Système de planification en mixed-initiative pour l'assistance à la gestion des systèmes informatisés complexes

Abdeldjalil Ramoul

▶ **To cite this version:**

Abdeldjalil Ramoul. Système de planification en mixed-initiative pour l'assistance à la gestion des systèmes informatisés complexes. Robotique [cs.RO]. Université Grenoble Alpes, 2018. Français. NNT : 2018GREAM070 . tel-02080943

HAL Id: tel-02080943
https://theses.hal.science/tel-02080943

Submitted on 27 Mar 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Abdeldjalil RAMOUL

Thèse dirigée par **Sylvie PESTY**, Professeur, UGA
et codirigée par **Damien PELLIER**, MCF, UGA

préparée au sein du **Laboratoire d'Informatique de Grenoble**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Système de planification en mixed-initiative pour l'assistance à la gestion des systèmes informatisés complexes

Mixed-initiative planning system to assist the management of complex IT systems

Thèse soutenue publiquement le **28 novembre 2018**,
devant le jury composé de :

Madame SYLVIE PESTY
PROFESSEUR, UNIVERSITE GRENOBLE ALPES, Directeur de thèse
Madame AMAL EL FALLAH SEGHROUCHNI
PROFESSEUR, SORBONNE UNIVERSITES - PARIS, Rapporteur
Monsieur SAMIR AKNINE
PROFESSEUR, UNIVERSITE LYON 1, Rapporteur
Monsieur RACHID ALAMI
DIRECTEUR DE RECHERCHE, CNRS DELEGATION MIDI-PYRENEES, Président
Monsieur DAMIEN PELLIER
MAITRE DE CONFERENCES, UNIVERSITE GRENOBLE ALPES, Co-directeur de thèse

Remerciements

Je remercie toutes les personnes impliquées dans le projet au sein du laboratoire d'informatique de Grenoble, de l'entreprise Intrinsec et de l'Association Nationale de la Recherche et de la Technologie (ANRT).

Je voudrais particulièrement remercier mes superviseurs de thèse, Damien Pellier, Sylvie Pesty, et Humbert Fiorino pour leur perspicacité, leur soutien et leur partage de connaissances qui ont rendu l'aboutissement de cette thèse possible.

Je tien à remercier les membres de mon jury de soutenance de l'intérêt qu'ils ont porté à mon travail et d'avoir accepté de participer au jury de ma soutenance de doctorat.

Pour finir, je voudrais remercier ma famille pour son soutien et ses encouragements pendant toutes ces années, pour avoir toujours été là pour moi durant les bons et surtout les mauvais moments. Je voudrais également remercier mes amis et collègues pour leur compréhension et leur soutien tout au long de ces années.

Résumé

Le concept de systèmes informatisés complexes rassemble tous les systèmes constitués d'un grand nombre de composantes inter-connectées et gérées par ordinateur. La configuration et la gestion de ces systèmes passe par une multitude de tâches critiques à leur bon fonctionnement et leur évolution. La problématique de la mise en place et la maîtrise des procédures de gestion et de configuration de tels systèmes devient un point critique au vu de la complexité croissante et du besoin d'évolution de ces derniers. L'objectif de cette thèse est de répondre à cette problématique, à travers le développement d'un outil de planification en mixed-initiative pour la co-génération automatique d'actions de gestion et de configuration, indépendamment du domaine d'application. Dans cette perspective, nous développons « *Grounded Totally Ordered Hierarchical Planner* » (GTOHP), un planificateur automatique hiérarchique, en « *Hierarchical Task Network* » (HTN), qui présente des performances élevées nécessaires à une interaction en mixed-initiative. Pour cela nous proposons un algorithme d'instanciation et de simplification des problèmes de planification *HTN* qui réduit de manière très significative leur complexité et améliore de ce fait les performances des algorithmes de planification. Nous proposons aussi une extension au langage de définition des domaines de planification automatique *PDDL* afin de modéliser les connaissances des experts du domaine d'application sous forme de méthodes de décomposition des tâches qui serviront à guider l'algorithme de planification *HTN*. Ensuite, nous intégrons au planificateur *GTOHP* des mécanismes de récolte de statistiques et d'informations sur les résultats des tests réalisés lors de la recherche de plans et nous l'intégrons dans le système « *Mixed-Initiative Planner* » (MIP) qui fournit plusieurs fonctionnalités d'interaction en mixed-initiative. Nous démontrons les performances élevées du planificateur *GTOHP* et les apports de l'algorithme d'instanciation et de simplification en le comparant à un planificateur *HTN* de l'état de l'art à travers une série d'expérimentations sur des domaines de planification issues de la compétition internationale de planification automatique. Enfin, nous proposons des critères d'évaluation pour les systèmes en mixed-initiative qui servent de base à la discussion du système *MIP*.

Mots clés

Planification automatique ; planification HTN ; Planification en mixed-initiative ; Instanciation et simplification des problèmes de planification HTN ; systèmes d'aide à la gestion.

Abstract

The concept of complex IT systems includes all systems consisting of a large number of inter-connected and computer-managed components. The configuration and management of these systems involves a multitude of tasks that are critical to their proper functioning and their evolution. The problem of defining procedures for managing and configuring such systems becomes very critical in view of their increasing complexity and their rapid evolution. The aim of this thesis is to develop a mixed-initiative planning tool for the automatic co-generation of a set of management and configuration actions, regardless of the application domain. In this perspective, we develop GTOHP, a hierarchical automatic planner, with HTN, that present the high performance needed for a mixed-initiative interaction. We propose an algorithm for the instantiation and the simplification of *HTN* planning problems, which significantly reduces their complexity and improves the performance of the planning algorithms. We also propose an extension to the *« Planning Domain Definition Language »* (PDDL) in order to modelize the knowledge of domain experts in the form of tasks decomposition methods that will be used to guide the *HTN* planning algorithm. Then, we integrate some mechanisms to *GTOHP* for collecting statistics and information about the tests results carried out during the plans search and integrate them into the MIP which provides several features of mixed-initiative interaction. We demonstrate the high performance of the *GTOHP* planner and the contributions of the instantiation and simplification algorithm, by comparing them to a state-of-the-art HTN planner through a series of experiments on planning domains from the international planning competitions. Finally, we propose a panel of evaluation criteria of mixed-initiative systems that serve as a basis for the discussion about the performances and contributions of the MIP system.

Keywords

Automated planning; HTN planning; Mixed-inititaive planning; HTN problems instanciation and simplification; management support systems.

Table des matières

1 Introduction générale **1**

I Background **7**

2 Planification automatique **9**

2.1 Planification classique 10

2.2 PDDL 10

2.3 Définitions et concepts de la planification classique 11

2.3.1 Prédicats, état, et opérateurs 11

2.3.2 Domaines, problèmes, et solutions 13

2.4 Planification HTN 14

2.4.1 Deux catégories de planificateurs HTN 14

2.4.2 Applications de la planification HTN 16

2.5 HPDDL 17

2.6 Définitions et concepts de la planification HTN 17

2.6.1 Prédicats, opérateurs, méthodes et tâches 18

2.6.2 Domaines, problèmes et solutions HTN 20

3 Planification en Mixed-initiative **22**

3.1 Planification en mixed-initiative 23

3.1.1 Le concept de mixed-initiative 24

3.1.1.A Approche 1 : L'initiative vue comme le contrôle sur le flux de conversation 24

3.1.1.B Approche 2 : L'initiative vue comme le contrôle sur la conduite de tâches 26

3.1.1.C Approche 3 : L'initiative vue comme la prise de contrôle d'une conversation en présentant un but à achever 27

3.1.1.D Approche 4 : L'initiative vue comme la prise de contrôle de la conversation ou de la conduite de tâches 27

3.1.2 Le rôle de l'humain dans les systèmes en mixed-initiative 29

3.2 Critères d'évaluation des systèmes de planification en mixed-initiative 29

3.3 Présentation des systèmes de planification en mixed-initiative 37
3.3.1 PSM/TRIPS 37
3.3.2 PASSAT 39
3.3.3 HICAP 40
3.3.4 MAPGEN 40
3.3.5 Système en mixed-initiative pour le guidage d'UAVs 42
3.3.6 Gtrans 43
3.3.7 CMMD 44
3.3.8 Autres systèmes 45

II Un système de planification en mixed-initiative fondé sur une approche de planification HTN totalement instanciée 47

4 GTOHP : Un planificateur HTN avec des problèmes instanciés et simplifiés 49
4.1 La complexité de l'instanciation des problèmes HTN 50
4.2 Algorithme d'instanciation et de simplification des problèmes HTN 52
4.2.1 Calcul des inerties 52
4.2.2 Instanciation et simplification des opérateurs 54
4.2.3 Instanciation et simplification des méthodes 57
4.3 Algorithme de planification HTN avec des problèmes instanciés et simplifiés 61
4.4 Expérimentations de l'algorithme d'instanciation et de simplification 62
4.4.1 Cadre expérimental 63
4.4.1.A Matériel 64
4.4.1.B Critères d'évaluation 64
4.4.2 Évaluation du taux de simplification 66
4.4.3 Évaluation du temps de calcul 70
4.4.4 Évaluation de la longueur des plans 76

5 MIP : Un système de planification en mixed-initiative 81
5.1 Architecture du système 82
5.1.1 Gestionnaire d'interaction 83
5.1.1.A Assistant de chargement des données 84
5.1.1.B Gestionnaire d'interrogation 85
5.1.1.C Gestionnaire de modifications 86
5.1.1.D Gestionnaire de planification 87
5.1.2 Gestionnaire de base de connaissances 88
5.1.2.A Avantages d'une représentation multi-niveaux des connaissances 88

5.1.2.B Données de haut-niveau . . . 89
5.1.2.C Données de bas-niveau . . . 90
5.1.3 Module d'instanciation et de simplification . . . 90
5.1.3.A Compilateur du problème . . . 91
5.1.3.B Calculateur d'inerties . . . 92
5.1.3.C Instanciateur et simplificateur . . . 92
5.1.4 Module de planification . . . 93
5.1.4.A Mécanisme de méta analyse de la planification . . . 94
5.1.4.B Mécanisme d'analyse des solutions potentielles . . . 94
5.1.4.C Mécanisme de planification any-time . . . 95
5.1.4.D Mécanisme de comparaison de plans . . . 97
5.1.4.E Mécanisme d'interaction continue . . . 97
5.2 Fonctionnalités de lecture des connaissances . . . 98
5.2.1 Interaction sur deux niveaux d'instanciation et de simplification . . . 98
5.2.2 Le listing des structures du problème . . . 100
5.2.3 Le getting des structures du problème . . . 103
5.3 Fonctionnalités de modification des connaissances . . . 106
5.3.1 Ajout, suppression et mise à jour des connaissances . . . 106
5.3.2 Modification du problème et raffinement du but . . . 110
5.4 Fonctionnalités de co-construction de plan . . . 112
5.4.1 Fonctionnalité d'incitation au changement . . . 112
5.4.2 Fonctionnalité de décomposition pas à pas . . . 113
5.4.3 Fonctionnalité de réponse immédiate . . . 113
5.4.4 Fonctionnalité d'amélioration des plans . . . 114
5.4.5 Explication du raisonnement . . . 114
5.4.6 Fonctionnalité d'explication de l'échec . . . 115
5.4.6.A Définition de l'échec . . . 115
5.4.6.B Réponse à l'échec . . . 116
5.4.6.C Formalisation de l'explication de l'échec . . . 116
5.5 Évaluation du système MIP . . . 117
5.5.1 Évaluation du guidage . . . 117
5.5.2 Évaluation du contrôle et de la charge de travail . . . 119
5.5.3 Évaluation de l'adaptabilité . . . 121
5.5.4 Évaluation de la gestion des erreurs . . . 122
5.5.5 Évaluation de la cohérence . . . 124

5.5.6 Évaluation technique 125

6 Conclusion générale **129**

Table des figures

2.1 Schéma simplifié du domaine *Rover*. 12

3.1 Architecture générique d'un système de planification en mixed-initiative. 24
3.2 Comparaison de la prise d'initiative dans l'exemple 2 selon les différentes définitions. . 28
3.3 Critères d'évaluation des systèmes en mixed-initiative. 30
3.4 Architecture d'un agent collaboratif de dialogue. 38

4.1 Nombre d'instances de méthodes du domaine *Rover* sans simplification 51
4.2 Vue globale de l'algorithme d'instanciation et de simplification des problèmes *HTN*. . . 53
4.3 Vue globale de l'algorithme d'instanciation et de simplification des opérateurs. 55
4.4 Vue globale de l'algorithme d'instanciation et de simplification des méthodes. 58
4.5 Graphiques représentant le nombre d'instances et de décompositions des domaines : (a) Rover, (b) Childsnack, (c) Satellite et (d) Barman sur une échelle logarithmique . . . 67
4.6 Graphiques représentant le nombre d'instances et de décompositions des domaines : (a) Depots, (b) Elevator, (c) Mystery et (d) Zenotravel sur une échelle logarithmique . . 68
4.7 Graphiques représentant le nombre d'instances et de décompositions des domaines : (a) Blocksworld, (b) Gripper, (c) Hiking, (d) Parking sur une échelle logarithmique . . . 69
4.8 Graphiques représentant le temps de traitement nécessaire afin trouver un plan solution pour les problèmes des domaines : (a) Rover, (b) Childsnack, (c) Satellite et (d) Barman 71
4.9 Graphiques représentant le temps de traitement nécessaire afin trouver un plan solution pour les problèmes des domaines : (a) Depots, (b) Elevator, (c) Mystery et (d) Zenotravel 72
4.10 Graphiques représentant le temps de traitement nécessaire afin trouver un plan solution pour les problèmes des domaines : (a) Blocksworld, (b) Gripper, (c) Hiking et (d) Parking 73

4.11 Graphiques représentant la longueur des premiers plans trouvé par les planificateurs *GTOHP etSHOP* pour les problèmes des domaines : (a) Rover, (b) Childsnack, (c) Satellite et (d) Barman 77
4.12 Graphiques représentant la longueur des premiers plans trouvé par les planificateurs *GTOHP etSHOP* pour les problèmes des domaines : (a) Depots, (b) Elevator, (c) Mystery et (d) Zenotravel 78
4.13 Graphiques représentant la longueur des premiers plans trouvé par les planificateurs *GTOHP etSHOP* pour les problèmes des domaines : (a) Blocksworld, (b) Gripper, (c) Hiking et (d) Parking 79

5.1 Architecture du système de planification en mixed-initiative MIP. 83
5.2 Processus de gestion des requêtes d'interrogation. 85
5.3 Processus de gestion des requêtes de modification. 86
5.4 Processus de gestion des requêtes de planification. 87
5.5 Processus de gestion de la base de connaissances. 89
5.6 Organisation des sous-modules impliqués dans le processus d'instanciation et de simplification. 91
5.7 Fonctionnement interne du module de planification. 93
5.8 Exemple typique d'échange utilisant la fonctionnalité de listing. 102
5.9 Exemple d'échange utilisant la fonctionnalité de getting. 105
5.10 Exemple d'échange utilisant les fonctionnalités de modification du domaine. 109
5.11 Exemple d'échange utilisant les fonctionnalités de modification du problème. 111
5.12 Diagramme résumant le degré de satisfaction du système *MIP* des critères d'évaluation des système de planification en mixed-initiative 127

Liste des tableaux

4.1 Exemple de table d'inertie du domaine *Rover* 54
4.2 Scores des planificateurs *GTOHP, SHOP* et *Fast-Downward* selon l'agile track de l'IPC dans les domaines *Rover, Childsnack,Satellite* et *Barman* 75
4.3 Scores des planificateurs *GTOHP* et *SHOP* selon l'agile track de l'IPC dans les domaines *Blocksworld, Depots, Elevator, Gripper, Hiking, Mystery, Parking et Zenotravel* 76

5.1 Ensemble des commandes permettant le listing des structures de la base de connaissance 101
5.2 Ensemble des commandes permettant le getting des structures de la base de connaissance 104
5.3 Ensemble des commandes de modification des structures de la base de connaissance . 107
5.4 Ensemble des commandes de modification du problème de planification 110
5.5 Fonctionnalités du système *MIP* répondant aux critères de guidage 118
5.6 Fonctionnalités du système *MIP* répondant aux critères de contrôle et de charge de travail 120
5.7 Fonctionnalités du système *MIP* répondant aux critères d'adaptabilité 121
5.8 Fonctionnalités du système *MIP* répondant aux critères de gestion des erreurs 123
5.9 Fonctionnalités du système *MIP* répondant aux critères de cohérence 125
5.10 Résumé des caractéristiques techniques du système *MIP* 126

Acronymes

ADL *« Action Description Language »* 10
APGEN *« Activity Plan Generator »* 41
BDI *« Belief Desire Intention executive »* 43
CMMD *« Coordinated Multi-source Maintenance on Demand »* 44
EUROPA *« Extensible Universal Remote Operations Planning Architecture »* 41
GTOHP *« Grounded Totally Ordered Hierarchical Planner »* iii
GTrans *« Goal TRANSformation »* 43
HATP *« Human Aware Task Planner »* 43
HICAP *« Hierarchical Interactive Case-Based Architecture for Planning »* 40
HPDDL « Hierarchical Planning Domain Defition Language » 17
HTE *« Hierarchical Tasks Editor »* 40
HTN *« Hierarchical Task Network »* iii
IHM *« Interaction Humain Machine »* 5
ISS *« International Space Station »* 44
MAPGEN *« Mixed-Initiative Activity Planning Generator »* 40
MI-CBP *« Mixed-InitiativeCase-Based Planning system »* 45
MIP *« Mixed-Initiative Planner »* iii
MRS *« Multi-Robot Supervisory System »* 43
NEO *« Noncombatant Evacuation Operations »* 40
NOAH *« Nets of Action Hierarchies »* 14
Nonline *« Non-linear Planner »* 15
PASSAT *« Plan-Authoring System based on Sketches, Advice, and Templates »* 39
PDDL *« Planning Domain Definition Language »* v
POP *« Partial Order Plan »* 45
PSM *« Planning Solving Manager »* 37

RSS *« Robot Supervisory System »* 43
SHOP *« Simple Hierarchical Ordered Planner »* 15
SIPE *« System for Interactive Planning and Execution »* 15
STRIPS *« Stanford Research Institute Problem Solver »* 10
TRIPS *« the Rochester Interactive Planner System »* 37
UAV *« Mixed-initiative planning and execution for unmanned aerial vehicles »* 42
UMCP *« Universal Method Composition Planner »* 15

1

Introduction générale

Nous observons ces dernières années une croissance importante dans le développement, la production et la mise en service des systèmes informatisés complexes tels que des centres de données, des infrastructures de production industrielle, des systèmes centralisés de gestion des transports en commun, etc. Ces systèmes informatisés nécessitent une infrastructure matérielle et logicielle qui doit suivre et même anticiper un tel développement. Parallèlement au développement des systèmes informatisés, la connectivité entre les différents terminaux à travers des réseaux privés ou sur internet devient incontournable induisant un changement radical dans la conception des logiciels et des infrastructures matérielles. Ce changement tend vers un hébergement distant et virtualisé des logiciels fournissant ces services. C'est ce qu'on appelle communément le *cloud*. Par exemple les infrastructures fournissant des services de type *cloud computing*, dont la taille et la complexité explosent ces dernières années, voient leur gestion et leur administration devenir de plus en plus difficile.

Le travail réalisé durant cette thèse s'inscrit dans le cadre d'une coopération entre le laboratoire d'informatique de Grenoble et la société *Cloud temple* (anciennement *Intrinsec)*, une entreprise française spécialisée dans l'hébergement et l'infogérence cloud des systèmes d'information critiques. Les ingénieurs et les techniciens de l'entreprise doivent intervenir sur des serveurs et des liaisons réseaux physiques réparties sur plusieurs centres de données, ainsi que sur la configuration des machines virtuelles, de la gestion du stockage, le routage du réseau, etc.

La réalisation de tâches de configuration ou d'administration sur des infrastructures de type *cloud*, et plus généralement sur des systèmes informatisés complexes, nécessite des interventions coordonnées sur plusieurs composants de l'infrastructure. Ces interventions suivent des procédures spécifiques à chaque tâche et nécessite au préalable, la mise en place d'une documentation qui doit être la plus exhaustive possible. Cependant une telle documentation comporte généralement une très grande quantité d'informations et nécessite un certain délai afin de trouver la procédure adéquate. Cette situation peut s'avérer être très problématique, surtout lors d'une intervention d'urgence où chaque minute perdue peut avoir des conséquences très graves sur le système. De plus, une telle documentation ne permet pas de s'adapter à l'évolution et aux changements de l'état du système et nécessite une description complète et exhaustive de toutes les procédures à réaliser pour toutes les situations possibles. Dans des systèmes complexes qui ne sont généralement pas bien formalisés, cette tâche est quasiment impossible et nécessiterait des ressources matérielles et humaines très importantes.

Face à la vitesse d'évolution des systèmes et aux limites d'une documentation classique, le développement d'outils d'aide à la gestion et à la configuration, capables de seconder les administrateurs et les utilisateurs dans leurs tâches d'exploitation devient une nécessité. La réalisation de tels outils nécessite d'adresser plusieurs types de verrous scientifiques et technologiques :

1. La génération automatique d'actions : Un outil d'aide à la gestion et à la configuration fournit à l'utilisateur une ou plusieurs procédures à suivre afin de réaliser une tâche donnée. Une procédure est présentée sous forme d'une suite d'actions que l'utilisateur doit appliquer dans un ordre précis. Face à la difficulté de produire manuellement toutes les procédures possibles, l'utilisation de la *planification automatique* permet de répondre à cette problématique, en générant automatiquement, à partir d'un état initial du système, une suite d'actions à appliquer afin d'atteindre un état but. La planification automatique utilise une définition formelle du domaine d'application sous forme de propriétés du système permettant de représenter son état, ainsi que des actions atomiques qui comportent des pré-conditions d'application et des effets sur l'état du système. Le défi ici est d'avoir une technique de planification automatique qui permet d'avoir des structures de contrôle d'un haut niveau d'abstraction afin de capturer au mieux l'expérience et les connaissances de l'utilisateur.

2. La formalisation des connaissances : L'utilisation de la planification automatique dans l'aide à la gestion et à la configuration des systèmes informatisés complexes nécessite la définition de centaines d'actions atomiques possibles dans ces systèmes et produit potentiellement des procédures, qu'on appelle des *plans*, contenant des centaines d'actions. Cette représentation très bas niveau des actions et de l'état du système induit une difficulté de compréhension et de formalisation des connaissances de la part des utilisateurs. Il faut donc, d'une part, que les utilisateurs de l'outil aient la possibilité de définir des tâches avec différents niveaux d'abstraction quant à la gestion et la configuration du système ; et d'autre part que l'outil soit capable de produire des plans suffisamment expressifs avec plusieurs niveaux de détails. Par exemple, Un plan d'une centaine d'actions peut être présenté à l'utilisateur sous forme d'une dizaines de tâches de haut niveau qu'il pourra dépiler jusqu'à atteindre les actions atomiques. Nous soutenons ici l'idée que l'utilisation d'une technique de planification hiérarchique sera plus à même de répondre aux besoins des utilisateurs, puisqu'elle permet d'exploiter les connaissances des experts du domaine dans la définition de la hiérarchie des tâches, et se rapproche plus de l'organisation hiérarchique naturelle qu'utilise les humains pour organiser leurs actions du quotidien.

3. L'acceptabilité et la compréhension des solutions : Face à l'aspect critique que peuvent avoir les actions réalisées par les utilisateurs de l'outil d'assistance sur le système, ces derniers doivent avoir pleine confiance dans la solution qui leur est proposée. Ils doivent ainsi être capables de : suivre les raisonnements de l'outil qui ont conduit à cette solution afin de se construire eux mêmes leur propre raisonnement et de comprendre la solution qui leur est proposée. En plus, si aucune solution ne leur est proposée, ils doivent être capables de comprendre l'origine de cet échec afin de pouvoir y remédier, en modifiant la description de l'état du système

ou des objectifs exprimés, ou en mettant à jour les connaissances sur le domaine d'application. Afin de répondre à ces besoins, nous proposons de suivre un schéma de planification en mixed-initiative où l'utilisateur et l'outil interagissent continuellement afin de générer un plan. Cette interaction se déroule dans les deux sens et permet, d'un coté, à l'utilisateur de suivre la recherche de solution et d'interroger le système afin d'être en mesure de comprendre son raisonnement et la solution qui en découle. De l'autre coté, elle propose d'intégrer à l'outil des mécanismes d'interaction qui lui permettent d'expliquer certains résultats et même de proposer des modifications des données d'entrée afin de trouver une solution.

L'objectif durant cette thèse est de développer un planificateur qui exploite au mieux les connaissances des experts des domaines d'application et de développer par la suite un outil d'aide à l'administration et à la configuration des systèmes informatisés complexes. Afin de réaliser cela, les travaux de recherche réalisés s'articulent sur deux axes complémentaires :

Planification HTN : En premier lieu, nous cherchons à répondre à la problématique de génération automatique de solutions, à travers le développement d'un planificateur automatique qui exploite au mieux les connaissances des experts et qui permet de les organiser sous forme hiérarchique. Pour cela, la planification HTN est l'approche la plus appropriée pour réaliser cela, puisqu'elle permet de définir une hiérarchie de tâches qui peuvent être décomposées en sous tâches ou être réalisée par une action primitive. Contrairement aux autres techniques de planification automatique, elle permet d'exploiter les connaissances des experts du domaine qui mettent en place cette hiérarchie en définissant la méthode avec laquelle une tâche est décomposée et les contraintes qui y sont liées. La planification HTN a été utilisée dans plusieurs domaines, comme la robotique, la composition de Web services, ou encore, les jeux vidéo. Cependant les algorithmes HTN restent très dépendants de la définition des méthodes de décomposition des tâches et souffrent d'un problème de performance dû en grande partie au facteur de branchement important de l'espace de recherche.

Notre contribution dans ce domaine consiste à proposer un algorithme d'instanciation et de simplification qui réduit la taille et la complexité des problèmes de planification HTN, en supprimant des actions et des méthodes de décomposition qui ne peuvent jamais être appliquées dans ce problème. Cette étape préliminaire permet de réduire l'espace de recherche et d'accélérer la procédure de recherche de solution. L'algorithme calcule, en amont, toutes les instances des structures de base dont a besoin la procédure de recherche, ce qui permet aussi d'accélérer la recherche de solution. Nous proposons aussi une extension hiérarchique au langage PDDL qui est le langage de référence dans la planification automatique. Cette dernière offre la possibilité d'étendre des problèmes déjà définis en rajoutant de la connaissance sous forme de mé-

thodes de décompositions. On peut alors avoir le choix d'utiliser un algorithme de planification classique ou différents types d'algorithmes hiérarchiques. Pour finir, nous introduisons le planificateur GTOHP qui prend en entrée un problème HTN complètement instancié et simplifié et démontrons son efficacité par rapport à un planificateur HTN équivalent.

Interaction en mixed-initiative : En second lieu, nous nous intéressons à l'interaction avec l'utilisateur ainsi qu'à la compréhension et l'acceptabilité des solutions qui lui sont proposées. Les tâches de configuration ou de gestion des systèmes informatisés complexes sont réalisées par différents profils d'utilisateurs : des utilisateurs experts du domaine d'application qui ont besoin d'assistance pour gérer la charge de travail, et des utilisateurs moins expérimentés qui ont besoin de suivre une procédure afin de réaliser une tâche précise. Cela pose bien sûr plusieurs problèmes tels que l'acceptabilité de la solution par les utilisateurs, l'organisation des échanges lors de la recherche, ou encore l'intégration continue des connaissances des experts.

Notre contribution dans ce domaine consiste à présenter un système de planification en mixed-initiative qui exploite la capacité d'abstraction et de raisonnement de l'utilisateur ainsi que la rapidité de calcul de la machine et sa capacité de prendre en charge une grande quantité d'information. Pour cela, nous intégrons au planificateur *GTOHP* des mécanismes de récolte de statistiques et d'informations sur les résultats des tests réalisés lors de la recherche de plans. Ensuite, nous l'intégrons dans le système MIP qui fournit plusieurs fonctionnalités d'interaction en mixed-initiative, comme des fonctionnalités de lecture et de modification des connaissances, de décomposition pas à pas, d'incitation au changement, d'explication de l'échec, etc. Nous évaluons, les capacités d'interaction en mixed-initiative du système à travers plusieurs critères, issus de l'étude de l'ergonomie en « *Interaction Humain Machine* » (IHM), que nous adaptons à la problématique de la planification en mixed-initiative

Ce manuscrit est organisé en deux parties : La première partie, composée de deux chapitres, dresse un état de l'art de la planification automatique HTN et en mixed-initiative. La deuxième partie, aussi composée de deux chapitres, présente le planificateur *GTOHP*, l'algorithme d'instanciation et de simplification des domaines HTN, et le système de planification en mixed initiative *MIP*.

— Le chapitre 2 présente un état de l'art de planification HTN et introduit le cadre théorique ainsi que les définitions formelles des structures et des notions de base qui sont utilisées tout au long de ce manuscrit.
— Le chapitre 3 introduit la notion d'initiative et définit de façon formelle les notions d'interaction et de planification en mixed-initiative. Il présente aussi un état de l'art des systèmes de planification en mixed-initiative, et détaille la grille d'évaluation que nous proposons.

— Le chapitre 4 aborde la complexité liée à l'instanciation des problèmes de planification automatique en général et ceux de la planification HTN an particulier. Il présente, en suite, l'algorithme d'instanciation et de simplification des problèmes HTN ainsi que l'algorithme de planification *GTOHP*. Pour finir, il présente une évaluation de ces derniers en fonction de critères de performance sur un des problèmes issus de la compétition internationale de planification.
— Le chapitre 5 présente l'architecture du système *MIP*, ces différents modules, ainsi que les fonctionnalités de mixed-initiative qu'il implémente. Il présente ensuite, une évaluation du système en fonction des critères introduits dans le chapitre 3.
— Pour finir, le dernier chapitre de conclusion propose des pistes et des améliorations qui pourraient être apportées à ces travaux.

Première partie

Background

2

Planification automatique

Sommaire

2.1 Planification classique 10
2.2 PDDL 10
2.3 Définitions et concepts de la planification classique 11
2.4 Planification HTN 14
2.5 HPDDL 17
2.6 Définitions et concepts de la planification HTN 17

Introduction

La planification automatique exploite une définition formelle d'un domaine d'application et d'un problème qui peut être exprimé sous différentes formes, dans le but de produire un plan solution. Dans la première partie de ce chapitre, nous introduisons le langage formel de définition des problèmes de planification classique et donnons les définitions formelles des structures et notions de base qu'il utilise. Dans la seconde partie du chapitre, nous faisons un état de l'art de la planification *HTN* et donnons les définitions formelles des structures *HTN* que nous proposons sous forme d'une extension hiérarchique du langage de définition des domaines de planification classiques *PDDL* (*Planning Domain Definition Language*).

2.1 Planification classique

Une tâche de planification automatique consiste à chercher une séquence d'actions qui s'appliquent dans un ordre définit, à partir d'un état initial. L'application de chacune de ces actions transforme l'état actuel en un nouvel état, jusqu'à atteindre un état but qui remplit les conditions souhaitées.

2.2 PDDL

PDDL (McDermott et al. , 1998; Fox & Long, 2003; Kovacs, 2011) est le langage de référence pour la définition des domaines de planification classique. Il est notamment utilisé dans les compétitions internationales de planification (*IPC*) qui évaluent les performances des planificateurs de l'état de l'art sur un ensemble de problèmes de planification. *PDDL* a permis d'unifier plusieurs langages de définitions des problèmes de planifaication, i.e, *« Stanford Research Institute Problem Solver »* (STRIPS) Fikes & Nilsson (1971) ou son extension *« Action Description Language »* (ADL) (Pednault, 1988, 1989). *PDDL* a séparé la définition du domaine de celle du problème, ce qui a permis d'avoir une définition du domaine qui regroupe tous les concepts liés au domaine d'application et une multitude de définitions des problèmes qui sont liées à l'état initial du problème, aux objets utilisés et aux but défini.

2.3 Définitions et concepts de la planification classique

Dans cette section nous allons présenter les concepts de base utilisés dans la planification classique. Nous nous limitons aux concepts essentiels à la compréhension du manuscrit, pour plus de détails, voir (Ghallab et al. , 2004).

2.3.1 Prédicats, état, et opérateurs

Définition 2.1. *Un prédicat est une expression syntaxique du langage logique de la forme* $p(u_1,\ldots,u_k)$ *où* p *est le symbole du prédicat et* $u_1,\ldots,u_k$ *sont ses paramètres (ou arguments). Ces paramètres peuvent être des variables typées ou des constantes, appelées aussi objets. Les variables et les constantes possèdent un type qui peut hériter d'un ou plusieurs autres types. Ces types peuvent être représentés sous forme de prédicats unaires dont la valeur de vérité ne change pas d'un état à un autre. Un prédicat définit une ou plusieurs instances qu'on appelle communément des faits.*

Définition 2.2. *Un fait est une expression de la forme* $f(u_1,\ldots,u_k)$ *où* f *est le symbole du fait qui est le même que le symbole du prédicat qui le définit et* $u_1,\ldots,u_k$ *sont les arguments du fait qui ne contiennent que des constantes du même type que les variables du prédicat qui le définit.*

Définition 2.3. *Un état* s *est représenté par un ensemble fini de faits. Si un fait* $f \in s$*, alors* f *est vrai dans l'état* s*, sinon si* $f \notin s$*, alors* f *est faux dans* s*. L'ensemble* S *de tous les états possibles dans le domaine est aussi un ensemble fini.*

Définition 2.4. *Un opérateur est un tuple* $o = (nom(o), pre(o), eff(o))$*.* $nom(o)$ *est une expression syntaxique de la forme* $t(u_1,\ldots,u_k)$ *où* t *est le nom de l'opérateur et* $u_1,\ldots,u_k$ *sont ses paramètres.* $pre(o)$ *et* $eff(o)$ *sont des expressions logiques qui définissent respectivement les pré-conditions qui doivent être vérifiées pour appliquer l'opérateur et les effets qui sont les propositions logiques produites par l'opérateur. Les propositions négatives sont respectivement notées* $pre^-(o)$ *and* $eff^-(o)$*,* $pre^+(o)$ *and* $eff^+(o)$ *sinon.*

Une *action* est une instance de l'opérateur. Une action a est *applicable* dans un état s si $pre^+(a) \subseteq s$ et $pre^-(a) \cap s = \emptyset$. L'état s' résultant de l'application de a dans l'état s est définit comme suit :

$$s' = \gamma(s,a) = (s - eff^-(a)) \cup eff^+(a)$$

Exemple 2.3.1. *Le domaine Rover*

Afin d'illustrer ces définitions, considérons le domaine Rover qui est une version simplifiée de la mission d'exploration spatiale sur Mars, lancée par la NASA en 2003. Il a été utilisé, comme banchmark dans les compétitions internationales de planification en 2002 (Long & Fox, 2003) et 2006 (Dimopoulos et al. , 2006). Il traite du déplacement des robots, de l'échantillonnage des sols, de la prise de photos de cibles et de la transmission de données avec plusieurs robots. Notre choix s'est porté sur le domaine Rover parce qu'il est très connu mais aussi, en raison de la nature du problème de planification qui peut être décomposé en deux parties : L'exploration des chemins qui permet de tester la décomposition récursive, et la réalisation de procédures qui traite des choix successifs parmi de nombreuses décompositions possibles. Dans la définition du domaine, des robots (rovers) sont équipés d'un ensemble d'équipements et doivent traverser plusieurs zones appelées waypoints dans le but de collecter des échantillons de roche, de sol ou photographier une cible, et enfin transmettre les données à travers un lander. L'une des principales difficultés de ce domaine est que chaque rover est conçu pour un certain type de terrain et n'est pas capable de traverser tous les waypoints, ou de réaliser toutes les mesures. Par conséquent, tous les rovers ne sont pas en mesure d'effectuer toutes les missions et la génération de plan doit prendre en compte cette difficulté.

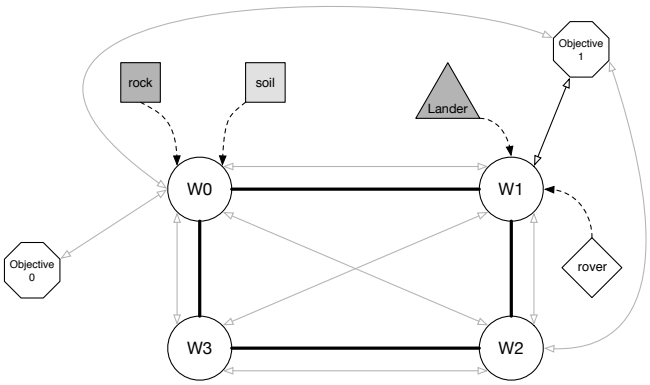


FIGURE 2.1 – Schéma simplifié du domaine *Rover*.

La figure 2.1 est une représentation partielle d'un problème simple de la compétition de planification IPC-5. *Les formes géométriques du schéma représentent les objets et les lignes qui les relient représentent des faits. w0, w1, w2, w3 sont des points de passage, només waypoints, et les lignes en gras indiquent que le rover peut traverser de l'un à l'autre. Une flèche avec des pointillés signifie que l'objet*

est au waypoint pointé, i.e. (at rover w1) ou (at_soil_sample w0). Les flèches doubles signifient que les objets sont visibles les uns par rapport aux autres, i.e. (visible w0 w2) ou (visible_from objective0 w0). Le but de ce problème est d'obtenir et de communiquer des données de sol et de roche à partir de w0, et d'obtenir des données d'image de l'objectif 1 et de les communiquer par l'intermédiaire du lander. Les propriétés de l'état but sont exprimés comme suit : (communicated_soil_data w0) AND (communicated_rock_data w0) AND (communicated_image_data objective1 low_res).

Exemple 2.3.2. *Exemple d'opérateur*

L'opérateur navigate définit en PDDL est comme suit :

```
(:action navigate
 :parameters (?x - rover ?from - waypoint ?to - waypoint)
 :precondition (and (available ?x) (can_traverse ?x ?from ?to)
                    (at ?x ?from) (visible ?from ?to))
 :effect (and (not (at ?x ?from)) (at ?x ?to)))
```

L'opérateur a trois paramètres : La variable ?x est du type "rover", ?from et ?to sont du type "waypoint". L'opérateur navigate définit le mouvement d'un rover d'un waypoint à un autre. Les pré-conditions de navigate sont que le rover ?x est disponible, qu'il est dans le waypoint ?from, que ?to est visible depuis ?from, et que ?x peut traverser depuis ?from pour aller à ?to. L'opérateur possède un seul effet positif qui est : (at ?x ?to), et un effet négatif qui est (not (at ?x ?from)). Donc en appliquant cet opérateur, le rover ?x ne sera plus dans ?from et sera dans ?to.

2.3.2 Domaines, problèmes, et solutions

En planification classique, les connaissances sur le domaine sont représentées sous forme de système de transition d'états où chaque état est représenté par un ensemble d'instances de prédicats dans lequel ils sont *vrais*. La notion de domaine de planification est séparée de celle du problème permettant de définir les connaissances statiques sur le domaine d'application indépendamment de l'état initial et du but des problèmes qui peuvent être définis sur ce domaine.

Définition 2.5. *Un domaine de planification classique est un tuple $D = (R, O, \gamma)$ où R est l'ensemble des prédicats définissant les propriétés du domaine, O est l'ensemble des opérateurs définissant les actions*

possibles, et γ est la fonction de transition d'état où $\gamma(s,a) = (s - eff^{-}(a)) \cup eff^{+}(a)$, si l'action $a \in A$ est applicable dans $s \in S$.

Définition 2.6. *Un problème de planification classique est un tuple $P = (D, s_0, g)$ où D est un domaine de planification classique, s_0 est l'état initial du problème et peut être n'importe quel état de S, et g est le but du problème. Il est représenté par un ensemble de faits. L'ensemble $S_g = s \in S \mid s\ satisfait\ g$ est l'ensemble des états du domaine qui contiennent les faits du but.*

2.4 Planification HTN

La planification en utilisant un HTN consiste, comme la planification classique, à trouver une suite d'actions qui s'applique à partir de l'état initial. Cependant, elle permet de rajouter de la connaissance au domaine de planifiation sous forme de décompositions de tâches hiérarchiques qui guide l'algorithme de recherche de plans en restreingnant l'espace de recherche. À l'inverse de la planification classique, la suite d'actions produites ne tends pas à produire un état but mais cherche plutôt à réaliser une tâche ou un ensemble de tâches tout en respectant les contraintes qu'elles induisent. Cette technique utilise un réseau de tâches hiérarchiques dont les nœuds sont des tâches composées ou primitives. Ces tâches sont reliées entres elles par des contraintes de différents types que nous aborderons plus en détails dans le formalisme de planification HTN.

L'utilisation d'une hiérarchie de tâches pouvant être décomposées en actions n'est pas une idée récente dans la communauté de planification automatique. Déjà en 1975, Sacerdoti (Sacerdoti, 1975a,b) proposait un planificateur appelé *« Nets of Action Hierarchies »* (NOAH) basé sur cette idée. Le système est construit dans une structure de données appelée *«procedure net»*. Cette structure de données introduit pour la première fois le concept de réseau et de méthodes. Ces deux concepts font aujourd'hui partie de tous les planificateurs HTN modernes. L'algorithme de planification NOAH fonctionne sur le principe de la décomposition d'un réseau de tâches initial en un réseau de tâches plus détaillé dans l'ordre défini par des contraintes d'ordre. À chaque décomposition, NOAH vérifie si le nouveau plan, ou réseau de tâches, obtenu est toujours correct, en calculant les conflits potentiels entre les tâches.

2.4.1 Deux catégories de planificateurs HTN

En fonction de la nature de l'espace de recherche exploré par les algorithmes de planification HTN pendant la recherche de plan solution, on peut diviser les planificateurs HTN en deux catégories (Georgievski & Aiello, 2015) : Les planificateurs avec un espace de recherche basé sur les plans, qu'on

va nommer *«espace de plans»*, et des planificateurs avec un espace de recherche basé sur les états qu'on va nommer *«espace d'états»*.

Dans la première catégorie, l'espace de recherche consiste en plusieurs réseaux de tâches représentant des plans partiels dans lesquels les planificateurs ne maintiennent pas l'état pendant la recherche. À chaque étape du processus de recherche, le réseau de tâches obtenu après une décomposition est considéré comme un plan partiel avec des contraintes qui doivent être satisfaites dans les décompositions suivantes. Le principal avantage de cette approche est de reporter les décisions concernant l'ordre des actions dans le plan à l'exécution alors que la planification dans un espace d'états produit généralement une séquence d'actions totalement ordonnée.

De nombreux travaux pionniers de cette catégorie ont suivi NOAH :

— « *Non-linear Planner* » (Nonline) développé par (Tate, 1976, 1977) étend l'approche de Sacerdoti et introduit un formalisme décrivant les domaines de manière hiérarchique.

— La planificateur *O-plan* (Currie & Tate, 1991) et son successeur *O-Plan2* (Tate et al., 1994) étendent le planificateur Nonline et proposaient pour la première fois des heuristiques de contrôle de recherche pour la planification HTN impliquant le typage des conditions ainsi que les contraintes de temps et de ressources.

— « *System for Interactive Planning and Execution* » (SIPE) (Wilkins, 1984) et son successeur *SIPE-2* (Wilkins, 1990) est un système de planification hiérarchique complet comprenant de nombreuses fonctionnalités telles que l'entrelacement de la planification et de l'exécution, le développement de plans interactifs, re-planification si des défaillances se produisent pendant l'exécution et possibilité d'utiliser des effets conditionnels dans les descriptions d'actions.

— « *Universal Method Composition Planner* » (UMCP) (Erol et al., 1994) est un planificateur HTN produisant des plans partiels avec une définition formelle des procédures de décomposition des tâches et de la réécriture des contraintes. Il est aussi le premier algorithme HTN dont l'exactitude et l'exhaustivité ont été prouvées.

En revanche, les planificateurs HTN récents et modernes tels que « *Simple Hierarchical Ordered Planner* » (SHOP) (Nau et al., 1999) maintiennent les états pendant le processus de recherche et explorent un espace d'états. Chaque réseau de tâches contient une représentation de l'état en plus des tâches. Une décomposition est applicable si et seulement si certaines contraintes exprimées comme pré-condition de la décomposition tiennent dans l'état. Ceci est très puissant car les conditions préalables permettent d'élaguer rapidement des décompositions peu prometteuses. Une comparaison complète de ces différents travaux est proposée dans (Georgievski & Aiello, 2015) et une analyse de complexité de la planification HTN est disponible dans (Erol et al., 1994; Alford et al., 2015). De nombreux travaux basés sur cette approche ont été développés au cours de la dernière décennie. Par

exemple, *SHOP* a été étendu afin de permettre de définir les sous-tâches d'une décomposition de façon partiellement ordonnée (Nau et al. , 2003). On peut citer aussi la production de plans temporels avec le planificateur *SIADEX* (de la Asunción et al. , 2005) ou des plans intégrant des préférences utilisateur (Sohrabi et al. , 2009). Il a également été étendu aux systèmes multi-agents (Pellier & Fiorino, 2007), à la recherche en ligne (Marthi et al. , 2008) ou à la résolution de problèmes de planification non déterministes Kuter et al. (2009). Le système de planification *GoDel* (Shivashankar et al. , 2013) combine la planification classique et HTN. Il définit un formalisme simple qui étend la planification classique pour inclure la décomposition des problèmes et un algorithme de planification basé sur ce formalisme. Il permet de combiner la recherche HTN, lorsque les décompositions sont disponibles, et la recherche prospective classique dans le cas contraire.

Enfin, (Amol & Subbarao, 1998) et (Pavel & Roman, 2005) proposent d'utiliser des solveurs *SAT* et *CSP* classiques en codant respectivement un problème de planification HTN dans un problème de satisfiabilité et un problème *CSP* dynamique. Ces deux techniques sont utilisées pour résoudre divers problèmes combinatoires et d'optimisation.

2.4.2 Applications de la planification HTN

En termes d'applications, HTN est l'une des techniques de planification les plus utilisées pour les applications du monde réel. Par exemple, il a été principalement utilisé pour la composition automatique de services Web. HTN est bien adapté à la composition automatique de services Web car les services Web sont décrits de manière hiérarchique comme le formalisme de planification HTN. Ainsi, la traduction peut être faite automatiquement. La première utilisation de la planification HTN pour la composition automatique des services Web a été faite par (Sirin et al. , 2004; Wu et al. , 2003). Dans ces travaux, les services Web sont automatiquement traduits en descriptions HTN en utilisant des ontologies, et le planificateur *SHOP2* est utilisé pour générer la séquence de services Web qui doit être exécutée sur le Web pour répondre à la requête d'un utilisateur. De nombreuses approches ultérieures ont été proposées : *OWL-S Xplan* (Klusch et al. , 2005) est une approche hybride qui combine la planification basée sur un graphe de plans (Blum & Furst, 1997) et la planification HTN pour fournir un plan avec des services Web incomplets, (Paik & Maruyama, 2007) combine la planification HTN et CSP pour planifier les contraintes des services web, (Chen et al. , 2009) propose d'associer le processus de décision Markovien et HTN afin de calculer plusieurs plans de composition, puis présenter les plus appropriés à l'utilisateur.

La planification automatique a été largement utilisée dans le domaine de la robotique, et la planification HTN ne fait pas exception. Par exemple, le planificateur HTN *HATP* (Lallement et al. , 2014) a

été spécialement conçu pour résoudre les problèmes de planification de la robotique en combinant la planification et le raisonnement géométrique. Dans (Hartanto, 2011; Belker et al. , 2003; Weser et al. , 2010), les techniques de planification HTN sont utilisées pour la navigation des robots, les manipulations des pinces de préhension, etc.

HTN a également été largement utilisé dans les outils d'aide à la décision. En effet, le raisonnement hiérarchique utilisé dans le planificateur HTN est proche de celui de l'humain. Les outils d'aide à la décision basés sur les techniques de planification HTN ont été appliqués à de nombreux domaines tels que la planification de missions de recherche et de sauvetage (Myers et al. , 2002, 2003; Allen & Ferguson, 2002; Aha et al. , 2001), dans l'exploration spatiale (Kichkaylo et al. , 2007) ou encore dans les systèmes de pilotage de drones (Bevacqua et al. , 2015). Nous aborderons ces systèmes plus en détails dans le prochain chapitre.

2.5 HPDDL

« Hierarchical Planning Domain Defition Language » (HPDDL) est une extension de la version 3.1 Kovacs (2011) du langage *PDDL*. Il s'inspire des notions d'expansions définies dans les premières versions de *PDDL* afin de permettre la définition des concepts de la planification HTN. Il permet notamment de rajouter, dans des domaines de planification classique définis en *PDDL 3.1*, de la connaissance sous forme de méthodes de décomposition. *HPDDL* permet aussi de définir des problèmes de planification HTN sous forme de tâches but à planifier et de contraintes à respecter en plus des notions déjà existantes, comme la définition des types, des objets, de l'état initial ou l'état but. Le but à travers cette extension est de définir un formalisme commun pour la définition des domaines de planification classique, HTN à espace d'état et HTN à espace de plans. Cela permet de définir directement une sur-couche HTN sur des problèmes de planification classique définis en *PDDL* qui existent déjà. Cela a permis notamment d'étendre des problèmes des compétitions de planification avec des connaissances hiérarchiques et de pouvoir évaluer les résultats d'un planificateur HTN sur ces derniers. *HPDDL* offre aussi la possibilité d'utiliser des planificateurs HTN à espace d'état et à espace de plans et de comparer les performances de ces deux approches à partir d'un même problème.

2.6 Définitions et concepts de la planification HTN

Cette section fournit les définitions de base pour introduire le langage de planification et les procédures de recherche HTN.

2.6.1 Prédicats, opérateurs, méthodes et tâches

Dans l'extension du langage *PDDL* que nous proposons, la définition des prédicats et des opérateurs est la même que dans la planification classique, avec la même syntaxe et la même contrainte de non duplication (cf. 2.3.1).

Définition 2.7. *Une méthode m est un triplé* $m = (nom(m), expansion(m), contraintes(m))$. $nom(m)$ *est une expression syntaxique de la forme* $t(u_1, \ldots, u_k)$ *où t est le nom de la méthode et* $u_1, \ldots, u_k$ *sont ses paramètres.* $expansion(m)$ *est la liste des sous-tâches obtenues en appliquant la méthode m à la tâche t.* $contraintes(m)$ *sont les contraintes rajoutées par la méthode m et qui doivent être vérifiées dans le plan solution.*

Quatre types de contraintes peuvent être définies dans une méthode :

— *Contraintes d'ordre* : une contrainte d'ordre est une expression de la forme *(series* $t_1, t_2, ..., t_k$*)*. Elle signifie que dans un plan solution, la tâche t_1 doit être ordonnée avant la tâche t_2, t_2 avant t_3 ainsi de suite jusqu'à t_k.

— *Contraintes « Before »* : Une contrainte du type *« Before »* est une expression de la forme *(before* (φ) $(t_1, ..., t_k)$*)* où φ est une expression logique, et $(t_1, ..., t_k)$ est une liste de tâches. Pour que cette contrainte soit vérifiée, l'expression φ est vraie avant l'exécution des tâches du groupe $(t_1, ..., t_k)$.

— *Contraintes « After »* : les contraintes du type *« After »* s'écrivent comme les contraintes *Before* sous la forme d'une expression *(after* (φ) $(t_1, ..., t_k)$*)*. Pour que cette contrainte soit vérifiée, φ doit être vraie après l'exécution de la dernière tâche de $(t_1, ..., t_k)$.

— *Contraintes « Between »* : Une contrainte *« Between »* est une expression de la forme *(between* (φ) $(t_{11}, ..., t_{1k})$ $(t_{21}, ..., t_{2k})$*)*. Pour que cette contrainte soit vérifiée, l'expression logique φ doit être vérifiée entre l'exécution de la dernière tâche de $(t_{11}, ..., t_{1k})$ et la première tâche de $(t_{21}, ..., t_{2k})$.

Nous avons choisi d'enrichir la définition des méthodes avec la possibilité de définir tous ces types de contraintes afin de couvrir tous les types de planifications HTN : La planification dans un espace d'états et la planification dans un espace de plans. Les contraintes d'ordre peuvent être définies dans les deux types de planification. Néanmoins, lors de la planification dans un espace d'états il n'y a nul besoin des autres contraintes à l'exception des contraintes du type *Before* qui seront considérées comme dés pré-conditions à l'application des décompositions d'une méthode. On a donc la possibilité de définir comme pour les opérateurs des pré-conditions $pre^-(m)$ et $pre^+(m)$ avec des contraintes *(before* (φ) $(t_1, ..., t_k)$*)* où $(\varphi) = pre^-(m) \cup pre^+(m)$.

Définition 2.8. *Une tâche est une expression syntaxique de la forme $t(u_1,\ldots,u_k)$ où t est le nom de la tâche et $u_1,\ldots,u_k$ sont ses paramètres. On nomme une instance de tâche toute tâche dont les paramètres sont soit des constantes, soit des instances de variables. Si la tâche t est syntaxiquement égale au nom d'une action, alors la tâche est dite primitive et est exécutée par cette action ; Sinon, si elle est syntaxiquement égale au nom d'une décomposition, alors la tâche est dite composée et elle est décomposable par cette décomposition. Si une tâche est primitive, elle peut être exécutée par une et une seule action mais si elle est composée, elle peut être décomposée en plusieurs décompositions.*

Les décompositions et les actions qui exécutent ou décomposent une tâche sont dits pertinents pour cette dernière.

Définition 2.9. *Une décomposition d est une instance d'une méthode qui ne contient que des constantes. Lors d'une planification dans un espace d'états, une décomposition d est applicable sur une tâche composée t et dans un état s si $nom(a) = t$ et $pre^+(d) \subseteq s$ et $pre^-(d) \cap s = \emptyset$. Lors d'une planification dans un espace de plans, une décompositions d est applicable sur une tâche composée t si $nom(a) = t$ et que les contraintes(d) n'introduisent pas d'incohérence dans le task-network (cf. définition 2.10).*

Exemple 2.6.1. *Exemple de méthode*

La méthode do_navigate du domaine Rover peut être définie dans HPDDL comme suit :

```
(:method do_navigate
 :parameters  (?x - rover ?from ?to - waypoint)
 :expansion   ((tag t1 (navigate ?x ?from ?mid))
               (tag t2 (visit ?mid))
               (tag t3 (do_navigate ?x ?mid ?to))
               (tag t4 (unvisited ?mid)))
 :constraints ((before (and (not(can_traverse ?x ?from ?to)) (not(visited ?mid))
                            (can_traverse ?x ?from ?mid)) t1)))
```

La méthode possède trois paramètres : Le paramètre $?x$ est du type "rover", $?from$ et $?to$ sont du type "waypoint". La méthode est applicable si (1) le rover $?x$ n'est pas capable de traverser directement du waypoint $?from$ au waypoint $?to$ (2) Il n'a pas encore visité le waypoint $?mid$ et finalement (3) Il doit être capable de traverser du waypoint $?from$ vers le waypoint $?mid$. L'application de la méthode implique d'exécuter de façon récursive la séquence de tâches : (navigate ?x ?from ?mid), (visit ?mid), (do_navigate ?x ?mid ?to) et (unvisited ?mid)). Il faut noter que contrairement aux opérateurs,

une méthode peut utiliser des variables qui n'ont pas été déclarées dans ses paramètres (comme $?mid$*, par exemple). Le type de ces variables doit être inféré avant le processus de planification.*

La définition d'un réseau de tâches hiérarchiques (ou *hierarchical task network*) est donnée comme suit :

Définition 2.10. *Un réseau de tâches hiérarchiques est une paire TN = (T,C), où T est l'ensemble des tâches et C l'ensemble des contraintes sur ces tâches (cf. définition 2.7)*

2.6.2 Domaines, problèmes et solutions HTN

Un domaine de planification HTN est une extension du domaine de la planification classique définissant les méthodes de décompositions des tâches en plus des opérateurs de la planification.

Définition 2.11. *Un domaine HTN est un tuple* $D = (R, O, M, \gamma)$ *où R est l'ensemble des prédicats définissant les propriétés du domaine, O est l'ensemble des opérateurs définissant les actions possibles, et M est l'ensemble des méthodes définissant les décompositions possibles des tâches.* γ *est la même fonction de transition d'état que dans la planification classique (cf. 2.5).*

Définition 2.12. *Un problème de planification HTN est un tuple* $P = (D, s_0, TN)$ *où D est un domaine de planification HTN,* s_0 *est l'état initial du problème, et* $Tn : (T, C)$ *est le réseau de tâches initial composé des tâches T définissant le but ainsi que les contraintes C qui portent sur elles.*

La solution totalement ordonnée d'un problème HTN $P = (D, s_0, TN)$ est un plan $\pi = \langle a_1, \ldots, a_n \rangle$ qui accomplit les tâches de TN tout en vérifiant ses contraintes. Dans un espace d'états, cela signifie qu'il existe des décompositions de tâches contenues dans T produisant π tel que π est exécutable à partir de s_0 et que chaque décomposition est applicable dans l'état qui précède sa première sous-tâche. La définition formelle peut être définit de manière récursive comme suit :

Définition 2.13. *Le plan* $\pi = \langle a_1, \ldots, a_n \rangle$ *est une solution pour le problème de planification HTN* $P = (D, s_0, TN)$ *dans les cas suivants :*

Cas 1 ; *TN contient un ensemble vide de tâches, alors le plan vide* $\pi = \langle\rangle$ *est la solution.*

Cas 2 ; *La première tâche* $t \in T$ *est primitive, alors* π *est une solution pour P s'il y a une action a définie par un opérateur* $o \in O$ *tel que (1) a pertinente t, (2) a est applicable dans* s_0 *et (3)* $\pi = \langle a_2, \ldots, a_n \rangle$ *est un plan solution pour le problème HTN :*

$$P' = (\gamma(s_0, a_1), T - \{t\}, O, M)$$

Cas 3. *Si la première tâche $t \in T$ est non-primitive, alors π est un plan solution s'il existe une décomposition d définie par une méthode $m \in M$ tel que d décompose t et π est une solution pour le problème de planification HTN :*

$$P' = (s_0, T - \{t\} \cup expansion(d), O, M)$$

Le plan solution d'un problème HTN peut être représenté sous forme de réseau de tâches primitif dont toutes les contraintes sont vérifiées. Cela permet de représenter les plans partiellement ordonnés résultant d'une planification dans un espace de plans.

Définition 2.14. *Un réseau de tâches primitif est un réseau de tâches particulier $TN_p = (T_p, C)$, où T_p est un ensemble de tâches primitives correspondant à des actions du domaine et C l'ensemble des contraintes sur ces tâches.*

Conclusion

Nous avons, dans ce chapitre, donné les définitions formelles des structures et notions de base de la planification classique, ainsi que de la planification *HTN* qui seront utilisées dans la suite de ce manuscrit. Nous avons vu que la planification *HTN* permettait, contrairement à la planification classique, d'exploiter des connaissances du domaine sous forme de méthodes de décomposition qui définissent une structure hiérarchique permettant de guider les algorithmes de planification dans la recherche de solution. Cette représentation hiérarchique des tâches se rapproche de la représentation naturelle. Cependant, ces techniques de planification sont vues par l'utilisateur comme une boite noire qui prend en entrée un domaine de planification et donne en sortie un plan solution ou un message d'erreur. De ce fait, l'utilisateur se retrouve écarté du processus de génération de solution, ce qui a de fortes chances de conduire un utilisateur non initié à l'incompréhension de la solution proposée et à terme au refus de l'outil de planification.

3

Planification en Mixed-initiative

Sommaire

3.1 Planification en mixed-initiative 23

3.2 Critères d'évaluation des systèmes de planification en mixed-initiative 29

3.3 Présentation des systèmes de planification en mixed-initiative 37

Introduction

Ce chapitre a pour but de répondre à la problématique de la compréhension et de l'acceptation par l'utilisateur des solutions qui lui sont proposées à travers l'introduction de la planification en mixed-initiative. Cette dernière exploite la capacité d'abstraction de l'humain et la capacité de calcul et de traitement de données des machines dans le but de fournir des plans qui soient facilement compris et acceptés par l'utilisateur. Dans ce chapitre, nous commençons par donner une définition formelle du concept d'interaction et de planification en mixed-initiative. Ensuite, nous proposons une grille d'évaluation des systèmes de planification en mixed-initiative qui servira de base à l'évaluation du système *MIP* qui est une des contributions de cette thèse. Enfin, nous faisons un état de l'art de la planification en mixed-initiative en présentant et analysant les systèmes qui ont apporté une contribution significative dans ce domaine.

3.1 Planification en mixed-initiative

Le premier constat qui peut être fait sur les planifications classique et HTN, est que la génération des plans, ainsi que leur qualité dépend directement des données saisies en entrée. Dans le cas ou ces dernières sont erronées, incomplètes ou ne correspondent pas vraiment à ce que recherche l'utilisateur, il lui sera très difficile de déceler l'anomalie et d'intervenir sur les données en entrée afin d'obtenir un plan satisfaisant. Le besoin d'un système d'assistance à la réalisation de tâches et de planification en mixed-initiative capable d'initier l'interaction avec l'utilisateur et de réagir aux modifications qu'il apporte est essentiel pour son acceptabilité et pour guider l'utilisateur dans sa compréhension du plan solution.

La planification en mixed-initiative aborde deux aspects de l'intelligence artificielle à savoir la planification et l'interaction. La figure 3.1 représente une architecture générique d'un système de planification en mixed-initiative. La majorité des systèmes de planification en mixed-initiative s'articulent autour d'un gestionnaire d'interaction et d'un module de planification. Le gestionnaire d'interaction gère tous les aspects de l'interaction avec l'utilisateur en terme de définition de but, de changements dans le domaine, de propositions, etc. L'interaction avec l'utilisateur peut se faire à travers plusieurs interfaces, graphiques, gestuelles, en langage naturel, etc. Le planificateur génère des plans solutions en fonction des connaissances du domaine, de l'état du problème et les spécifications apportées à travers le gestionnaire d'interaction. Le gestionnaire des données regroupe tous les outils qui permettent de représenter les connaissances du domaine ou l'état du problème.

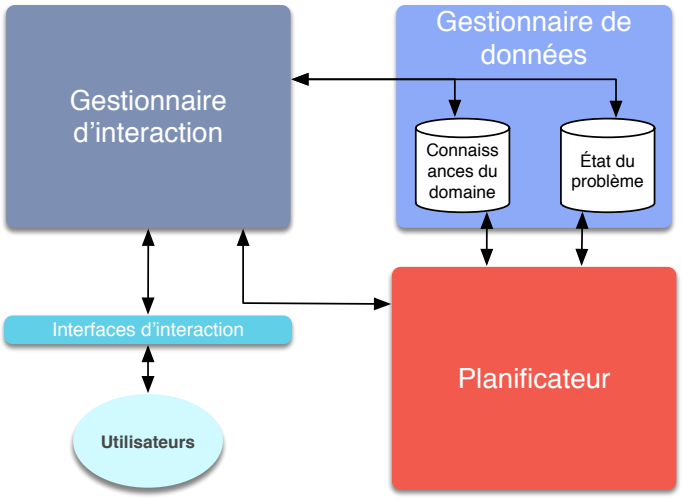


FIGURE 3.1 – Architecture générique d'un système de planification en mixed-initiative.

3.1.1 Le concept de mixed-initiative

Avant d'aborder plus en détails les systèmes de planification en mixed-initiative, on s'intéresse d'abord aux définitions données à l'initiative et la mixed-initiative dans le cadre de la coopération pour la résolution de problèmes. Le terme mixed-initiative a été cité pour la première fois en 1970 dans (Carbonell, 1970) qui présente 'SCHOLAR'(Carbonell & Collins, 1970), un système d'aide à l'apprentissage assisté par ordinateur permettant d'avoir un dialogue entre un humain et un ordinateur. En plus de répondre aux questions de l'humain, le système est capable, de formuler des questions, d'analyser la réponse de l'humain et d'agir en conséquent. Les termes "Initiative" ou "Mixed-Initiative" ont été ensuite largement utilisés dans les systèmes d'interaction mais leurs définitions ne font pas consensus (Novick & Sutton, 1997; Cohen et al. , 1998). L'initiative est alors définie en fonction de plusieurs approches, qu'elle soit liée au dialogue, à la réalisation des tâches ou à l'hybridation des deux.

3.1.1.A Approche 1 : L'initiative vue comme le contrôle sur le flux de conversation

Définition 3.1. *L'initiative dans un dialogue est la prise de contrôle de l'échange. (Walker & Whittaker, 1990)*

Dans cette approche, la prise de l'initiative est liée exclusivement aux modalités de l'échange conversationnel entre plusieurs agents. On dit alors que l'agent qui engage la conversation lors d'un

échange à travers une question ou une instruction nécessitant une action ou une réponse possède l'initiative. On peut dès lors définir la mixed-initiative comme suit :

Définition 3.2. *La mixed-initiative dans un dialogue est la possibilité d'avoir plusieurs agents capables, chacun à son tour de prendre le contrôle lors d'un échange.*

Les exemples suivants représentent des échanges donnés dans (Cohen et al. , 1998) et (Smith, 1993) reprenant une portion de dialogue avec un système en langage naturel pour l'assistance à la réparation des circuits électriques pour le premier et un assistant de planification de voyage en mixed-initiative pour le deuxième :

Exemple 1. *Assistant de circuit*

C : *This is the circuit fix it shop. How may I help you?*
U : *I want to fix a circuit.*
C : *What is the id of the circuit?*
U : *Rs111.*
C : *The LED is supposed to be displaying alternately flashing one and seven.*
U : *The LED is off.*
C : *What is the switch at when the LED is off?*
U : *The switch is down.*
C : *Put the switch up.*
U : *Okay.*

Dans cet exemple, l'initiative est exclusivement détenue par le système **"C :"** qui, à chaque étape pose une question ou donne une instruction à l'utilisateur **"U :"** qui ne fait que répondre ou valider la réalisation de la tâche demandée. Dans l'exemple 2, l'initiative bascule entre les deux agents. Dans les trois premiers échanges, l'initiative est détenue par l'utilisateur mais elle bascule vers le système lorsqu'il donne une information qui ne lui était pas demandée dans (4). L'initiative est ensuite reprise par l'utilisateur dans (5) lorsqu'il demande au système de lui donner le chemin d'Atlanta à Syracuse. Elle bascule ensuite vers le système en (6) qui demande une information et continue en la possession du système jusqu'à la fin du dialogue. À travers l'exemple 2 on peut voir un échange en mixed-initiative qui permet à tous les agents intervenants dans l'interaction de prendre l'initiative de l'échange.

Exemple 2. *Système de planification de transports en mixed-initiative (Ferguson et al. , 1996)*

U : *I need to get to Detroit, Syracuse and Montreal.*
C : *Ok. [highlights three cities]*

U : *Let's send the trains from Chicago to Detroit.*
C : *OK. [draws route] But the terminal at Toledo is delaying traffic due to bad weather. Trains will take an additional five hours to move through there.*
U : *That's ok. Now let's go from Atlanta up to Syracuse.*
C : *Please tell me a route to use to get from Atlanta to Syracuse.*
U : *Go via Charleston and Pittsburgh.*
C : *OK. But traffic through Cincinnati is being delayed due to construction.*

On voit à travers ces deux exemples qu'en considérant cette définition dans les systèmes d'interaction humain-machine, la mixed-initiative est exclusivement liée à la capacité de chaque agent à prendre l'initiative en posant une question, en demandant la réalisation d'une action ou en apportant une information qui n'est pas demandée.

3.1.1.B Approche 2 : L'initiative vue comme le contrôle sur la conduite de tâches

Définition 3.3. *L'initiative dans une tâche est l'acte de prendre le contrôle pour suggérer ou exécuter une tâche qui joue un rôle déterminant dans la résolution du problème(Cohen et al. , 1998).*

En d'autres termes, les agents dans cette définition prennent en main la direction de la recherche de solution soit en demandant des informations aux autres agents ou en affectant une tâche nécessaire à la résolution du problème à un ou plusieurs agents. L'agent qui a l'initiative de la tâche en cours ne la perd pas jusqu'à ce qu'elle bascule vers un autre agent qui prend l'initiative pour initier une tâche bloquante pour la tâche en cours ou s'il détecte une incohérence ou un manque d'information dans la tâche courante. Un agent peut s'affecter une tâche à lui même sans perdre l'initiative. Il faut bien faire la différence entre la *"task initiative"* qui indique qui est l'agent en charge de la réalisation d'une tâche, et la *"dialogue initiative"* qui indique l'agent qui dirige le dialogue actuel (Chu-Carroll & Brown, 1997). Même si des interactions (en terme de dialogue) peuvent se produire pendant la tâche courante, l'initiative ne bascule pas en fonction du dialogue mais plutôt en fonction des tâches. On peut alors définir la mixed-initiative comme suit :

Définition 3.4. *La mixed-initiative dans une tâche est la possibilité d'avoir plusieurs agents ayant la capacité de prendre alternativement le contrôle et de suggérer ou d'exécuter une tâche qui joue un rôle déterminant dans la résolution du problème.*

Reprenons l'échange de l'exemple 2 pour la planification de transports. Contrairement au cas de l'initiative de dialogue, l'initiative de tâche n'est pas prise par l'utilisateur dans la première phrase,

puisqu'il ne fait qu'énoncer un but. Elle est premièrement prise de la phrase 2 à 4 par le système qui tente de réaliser la tâche en proposant une solutions et en considérant les choix de l'utilisateur. Dans la phrase 5, la solution est validée par l'utilisateur marquant de ce fait la fin de la première tâche. L'initiative n'est alors plus prise par aucun des deux agents. Toujours dans la phrase 5, l'utilisateur énonce un autre but, qui est d'aller d'Atlanta à Cyracuse. Dans la phrase 6, l'initiative est prise par le système pour réaliser la tâche, mais n'ayant pas de solution, il demande cette information à l'utilisateur qui la lui fournit sans pour autant prendre l'initiative. L'initiative reste ainsi en possession du système jusqu'à la fin de la tâche.

3.1.1.C Approche 3 : L'initiative vue comme la prise de contrôle d'une conversation en présentant un but à achever

Définition 3.5. *L'initiative est la prise de contrôle de la conversation en faisant un énoncé qui présente un objectif à achever pour tous les participants(Cohen et al. , 1998).*

Cette définition de l'initiative est un mix des deux précédentes définitions puisqu'elle voit l'initiative comme la prise de contrôle de la conversation et en même temps de la conduite du processus de résolution. Contrairement à (Chu-Carroll & Brown, 1997), cette définition ne fait pas la distinction entre l'initiative de dialogue et celle des tâches. Si on reprend l'exemple 2 avec cette définition, l'utilisateur prend l'initiative lorsqu'il définit le but en (1) et la garde durant toute l'interaction pour réaliser ce but (2, 3 et 4). Il la reprend ensuite dans (5) pour définir un nouveau but et la garde jusqu'à la fin. Ne suivant que la définition du but pour le basculement de l'initiative et ne considérant pas (6) comme une définition de but mais plutôt une demande d'information, on se retrouve avec un échange qui n'est pas en *mixed-initiative* puisque le système ne prend jamais l'initiative de définir un but.

3.1.1.D Approche 4 : L'initiative vue comme la prise de contrôle de la conversation ou de la conduite de tâches

Partant du constat que la définition du but ne peut pas exprimer à elle seule la prise d'initiative et que la définition de l'initiative en prenant en compte le dialogue et la réalisation des tâches couvre mieux différents cas d'interaction, nous proposons une définition de l'initiative qui reprend celle de la troisième approche tout en élargissant les cas de prise d'initiative comme suit :

Définition 3.6. *L'initiative est la prise de contrôle de la conversation avec une assertion nécessitant une réaction des autres agents ou la réalisation d'une tâche qui joue un rôle déterminant dans la résolution*

du problème.

On considère alors qu'un agent a l'initiative quand il définit un objectif, il demande des informations répondants à un besoin, il apporte des informations pouvant aider dans la recherche de solutions, il prend en charge la réalisation de tâche ou quand il demande la réalisation d'une tâche nécessaire à la recherche de solution. Cette vision de l'initiative permet de mieux distinguer les points importants lors de la résolution de problèmes et de mieux mettre en avant l'apport de chaque agent lors de cette tâche. On peut alors définir la mixed- initiative comme suit :

Définition 3.7. *La mixed-initiative est la possibilité d'avoir plusieurs agents ayant la capacité de prendre le contrôle de la conversation avec une annonce nécessitant une réaction des autres agents ou de l'exécution des tâches par la prise en charge de la réalisation d'une tâche qui joue un rôle déterminant dans la résolution du problème.*

En reprenant l'exemple précédent, on a une prise de l'initiative par l'utilisateur dans les interaction (1 et 5) en définissant un but et dans (7) en donnant un trajet entre Atlanta et Syracuse, alors que le système possède l'initiative lors des autres échanges en prenant en charge la recherche et en proposant des solutions dans (2, 4 et 8), en attendant une réponse dans (3), et en demandant des informations nécessaires à la recherche. La figure 3.2 regroupe sous forme graphique la prise d'initiative dans l'exemple 2 selon ces quatre définitions. On voit que la définition 4 est celle qui détecte le plus de basculement d'initiative en pointant avec précision les apports de chacun des intervenants dans la résolution du problème.

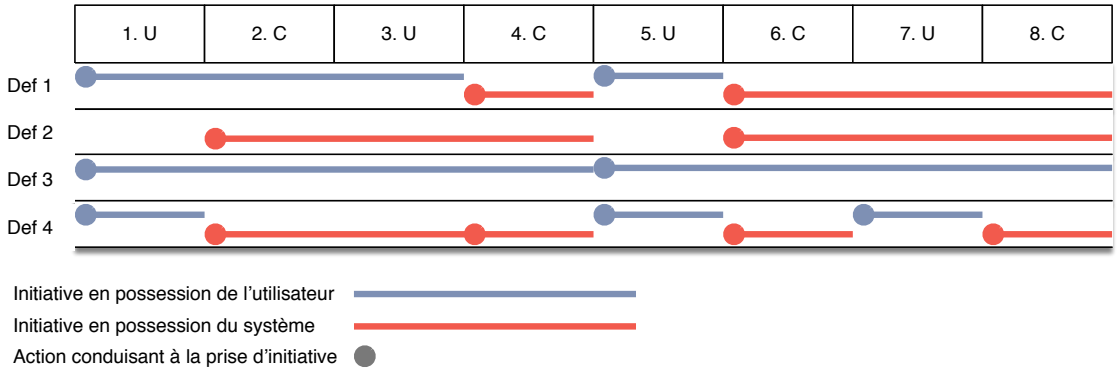


FIGURE 3.2 – Comparaison de la prise d'initiative dans l'exemple 2 selon les différentes définitions.

Après avoir définit la planification ainsi que l'initiative et la mixed-initiative séparément, ont peut alors poser la définition de la planification en mixed-initiative comme suit :

Définition 3.8. *La planification en mixed-initiative est la co-construction de plans à travers l'interaction d'agents ayants la capacité de prendre le contrôle de la conversation ou de l'exécution des tâches nécessaires au processus de planification.*

3.1.2 Le rôle de l'humain dans les systèmes en mixed-initiative

Comme mentionné précédemment, un système en mixed-initiative doit permettre et organiser l'interaction entre plusieurs agents apportant chacun une expertise complémentaire à celle des autres agents et travaillant en concert pour la réalisation d'une tâche donnée. Dans un système de planification en mixed-innitiative, cette expertise est généralement répartie entre un ou plusieurs agents de planification automatique et un ou plusieurs utilisateurs experts. Les machines apportent la vitesse de calcul et la capacité de manipulation de grandes quantités de données et les utilisateurs apportent une capacité d'abstraction et de guidage qui manque beaucoup aux machines avec les technologies actuelles. Les utilisateurs experts d'un domaine apportent aussi leur expertise à travers la définition de connaissances supplémentaires, l'apport d'informations complémentaires ou encore une analyse critique des connaissances précédemment fournies au système.

3.2 Critères d'évaluation des systèmes de planification en mixed-initiative

Malgré l'essor qu'ont connu les systèmes en mixed-initiative et les différents domaines d'application dans lesquels ils ont étés utilisés, on trouve très rarement des travaux présentant des critères précis pour leur évaluation. (Cortellessa & Cesta, 2005) et (Tecuci et al. , 2007) abordent les problématiques d'interaction auxquelles doivent répondre les systèmes d'assistance en mixed-initiative, i.e, la problématique des tâches qui évalue le degré de responsabilité de l'utilisateur ou du système dans la réalisation des différentes tâches, les problématiques du contrôle, de la compréhension, de la communication, de la personnalisation, de l'architecture ou encore la problématique de l'évaluation. Chaque point vise à placer l'initiative d'interaction sur une échelle de positionnement entre l'utilisateur et le système et essaye de définir quel degré est le mieux adapté pour une collaboration efficace.

Ces travaux nous apportent de nombreux points clés pour comparer et évaluer les systèmes d'assistance en mixed-initiative. Cependant, ils s'intéressent essentiellement aux problématiques d'interaction entre l'utilisateur et le système et ne sont pas assez détaillés dans le cadre d'une analyse des mécanismes de planification, de la compréhension de l'utilisateur et de l'acceptation du système. Ces derniers points étant très liés à la conception et l'ergonomie du système, il est apparu clairement qu'il

serait judicieux de se fonder sur les critères de (Scapin & Bastien, 1997) qui font consensus et restent une référence dans la conception et l'évaluation des systèmes interactifs tout en mettant en avant pour chaque critère d'ergonomie les propriétés vues précédemment avec celles que nous jugeons utiles pour l'évaluation des systèmes de planification en mixed-initiative. Il faut noter qu'en adaptant les critères d'ergonomie aux systèmes de planification en mixed-initiative, il est nécessaire de considérer toutes les interactions qui peuvent se produire dans le cadre de la planification automatique, et non pas se restreindre à une interaction homme-machine. Dans leurs travaux, Scapin et Bastien organisent les critères d'évaluation en huit grands critères qui se décomposent en sous-critères plus spécifiques. Dans le cadre de la planification en mixed-initiative, les critères de conception et d'évaluation que nous proposons sont présentés dans la figure 3.3 :

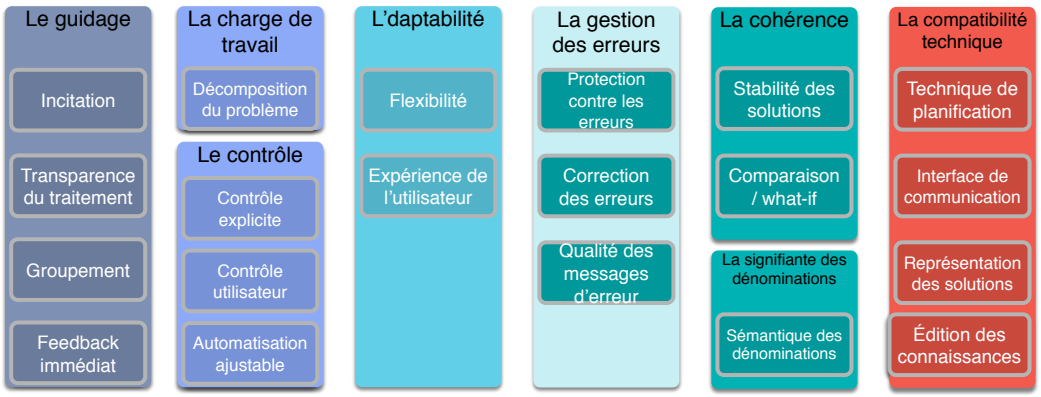


FIGURE 3.3 – Critères d'évaluation des systèmes en mixed-initiative.

Le guidage : Est l'ensemble des techniques et outils dont le système est pourvu afin de guider l'utilisateur lors de la modification du problème ou pour l'informer au mieux en lui fournissant des informations sur le problème et le processus de planification. Quatre sous-critères sont nécessaires pour le guidage de l'utilisateur dans un système de planification en mixed-initiative :

L'incitation : Est la capacité du système à conduire l'utilisateur à réaliser une action de planification ou de modification en détectant des améliorations, et en proposant des modifications au problème actuel, des alternatives aux objectifs définis ou le moyen de les atteindre. Dans Un système en mixed-initiative, l'agent responsable de la planification doit pouvoir prendre l'initiative pour demander des modifications en se fondant sur des évaluations en temps réel de la recherche de solution ainsi que sur les données collectées lors de recherches précédentes.

Exemple d'incitation : Proposer à un utilisateur de lever une contrainte sur le but car le système a pu trouver plusieurs solutions partielles lors de la planification qui ont été re-

jetées à cause de cette contrainte. Si l'utilisateur juge cette contrainte secondaire, il peut donc lever cette contrainte et suivre le plan généré par le système.

La transparence du traitement : Un système en mixed-initiative doit fournir des solutions et prendre des initiatives compréhensibles par l'utilisateur. Pour ce faire, le système doit implémenter des mécanismes permettant d'expliquer toutes les actions qu'il réalise lors de la planification ou l'interaction. Cette capacité passe par des modèles de raisonnement et de représentation des données compréhensibles par les utilisateurs et structurés de façon indépendante pour permettre d'avoir un système évolutif sans changer le modèle présenté à l'utilisateur.

Exemple de transparence de traitement : Fournir à la demande de l'utilisateur les étapes de vérification et de validation des pré-conditions d'une action ou les contraintes d'une méthode. Ou encore, mettre en avant les actions qui ont permis de produire un fait dans un état donné.

Le groupement : Est la capacité de fournir à l'utilisateur un moyen de regrouper les éléments des problèmes liés entre eux par une relation de composition, de génération, de consommation, etc. Contrairement à un utilisateur humain, un programme a la capacité de lister automatiquement un très grand nombre d'éléments en lien avec un ou plusieurs autres éléments du problème. Grouper les éléments permet aussi de voir rapidement les éléments qui seront affectés lors d'un changement sur l'élément qui les regroupe

Exemple de groupement : Donner la liste des actions produisant une propriété donnée, ou la liste des propriétés potentiellement produite lors de la réalisation d'une tâche.

Le feedback immédiat : La pierre angulaire des systèmes de planification en mixed-initiative est l'interaction, et cette dernière nécessite un temps de réponse beaucoup plus court par rapport à celui de la planification de certains problèmes. Faute d'un temps de réponse stable et acceptable, les systèmes en mixed-initiative doivent gérer ce problème en fournissant des mécanismes pour le contourner.

Exemple de feedback immédiat : Cette contrainte peut être gérée en fournissant une réponse positive (avec solution) ou négative (sans solution) après un temps défini même si le planificateur n'as pas encore trouvé de solution. Une autre façon de répondre à cette contrainte, consiste à utiliser une des capacités de guidage du système comme la transparence de traitement, avec un certain niveau d'abstraction, pour expliquer ce que le système est entrain de faire ; ou la capacité d'incitation pour proposer à l'utilisateur de réaliser des modifications pour accélérer la génération de solution.

La charge de travail (décomposition du problème) : Le critère de charge de travail s'intéresse aux techniques et outils mis en place afin d'aider l'utilisateur dans la compréhension et la manipu-

lation du problème en réduisant sa charge cognitive et en augmentant de ce fait l'efficacité de l'interaction. La charge de travail peut être décomposée en sous-critères :

La décomposition multi-niveaux du problème : La décomposition du problème en plusieurs niveaux permet de réduire la charge cognitive des tâches à réaliser. En général, dans les systèmes en mixed-initiative, le problème défini par l'utilisateur est, ou peut être, décomposé en plusieurs tâches ou objectifs permettant à l'utilisateur d'interagir avec le système à chaque niveau pour modifier, planifier ou simplement obtenir une représentation raffinée de la définition initiale du problème.

Exemple de décomposition multi-niveaux : La décomposition de l'état but à atteindre en sous états intermédiaires dont la planification peut être réalisée séparément. La décomposition d'une tâche à réaliser en sous tâches qui peuvent être planifiées individuellement.

Le contrôle : Le critère de contrôle s'intéresse aux mécanismes de partage de contrôle entre les différents agents des systèmes en mixed-initiative, et plus spécialement entre un opérateur humain et le système. Le critère de contrôle peut être décomposé en trois sous-critères :

Contrôle explicite : S'intéresse à la perception qu'ont les agents du contrôle sur les tâches en cours et à venir. Pour voir si le système fournit les informations nécessaires afin de permettre à un utilisateur de savoir si le traitement en cours est réalisé en réponse à une requête de sa part ou celle du système. Le système doit aussi spécifier clairement à l'utilisateur si une action de sa part est nécessaire ou s'il doit attendre une réponse.

Exemple de contrôle explicite : Annoncer textuellement ou en langage naturel le traitement qui va être réalisé et en réaction à quoi il a été réalisé.

Contrôle utilisateur : permet de dire si un utilisateur a toujours la main et peut prendre le contrôle du système, même s'il n'a pas l'initiative en cours. Un système en mixed-initiative offre la possibilité à plusieurs agents de prendre l'initiative et de réaliser des tâches ou en requérir une de la part d'autres agents. Dans le cadre d'une collaboration avec un humain, il est très important de lui donner le contrôle du système ou au moins du rythme de l'interaction et de la planification sous-peine de le voir rejeter le système et toutes les solutions qu'il propose

Exemple de contrôle utilisateur : Annuler ou interrompre un traitement en cours, demander une réponse ou une solution partielle immédiatement.

Automatisation ajustable : s'intéresse à la capacité du système en mixed-initiative à offrir différents niveaux d'automatisation de la planification. Il peut s'agir d'une automatisation binaire, cela signifie que le système offre à l'utilisateur la possibilité de choisir entre une solution générée automatiquement ou une solution qui est construite étape par étape

avec lui. L'automatisation peut aussi être ajustable avec des systèmes qui permettent d'automatiser une partie de la planification et de construire étape par étape une autre partie du plan en fonction des choix de l'utilisateur.

Exemple d'automatisation ajustable : Sur un ensemble de tâches, planifier une tâche étape par étape et laisser le système s'occuper des tâches restantes.

L'adaptabilité : Est la capacité du système à s'adapter aux besoins et aux préférences de utilisateur, ainsi qu'aux changements de contexte afin de lui proposer les solutions les plus proches de ses attentes. Trois sous-critères permettent d'évaluer l'adaptabilité d'un système de planification en mixed-initiative :

La flexibilité Est la capacité du système à fournir aux utilisateurs plusieurs façons ou chemins possibles pour atteindre leurs buts, que ça soit pour la planification ou pour la gestion des données. La qualité d'une solution n'est pas le seul critère à prendre en compte pour son acceptabilité. Afin de produire des solutions acceptables, il est très important de suivre le raisonnement de l'utilisateur et de lui fournir une solution qui s'approche le plus de ses attentes.

Exemple de flexibilité : Si un plan est trouvé et que l'utilisateur veut remplacer une sous-tâche par une autre ou un objet A par un autre du même type, le système s'adapte et fournit un nouveau plan incluant les changements souhaités.

Expérience de l'utilisateur : Les utilisateurs des systèmes de planifications en mixed-initiative peuvent avoir différents niveaux de connaissances en planification automatique ou dans le domaine d'application. L'interaction et les accès données aux utilisateurs doivent s'adapter à leurs niveaux. Le critère d'expérience s'intéresse aux mécanismes mis en place par les systèmes de planification en mixed-initiative permettant d'offrir à l'utilisateur les accès et l'interaction qui correspondent à son niveaux d'expertise.

Exemple d'adaptation à l'expérience de l'utilisateur : Adaptation des données du domaine ou du problème affiché en fonction du niveau de l'utilisateur. Restreindre et contrôler les modifications des données réalisées par l'utilisateur.

Gestion des erreurs ; concerne toutes les fonctionnalités implémentées par le système en mixed-initiative et permettant de gérer les erreurs qui peuvent se produire lors de la planification ou l'analyse des données. Elle se décompose en trois sous-critères :

La protection contre les erreurs ; s'intéresse aux capacités du système à éviter que des erreurs se produisent ou que des incohérences soient introduites lors de la planification, la définition ou la modification des données du domaine et du problème. Ce critère est très lié au guidage et à l'incitation puisqu'il permet de guider l'utilisateur pour ne pas commettre d'erreur ou introduire d'incohérences

Exemple de protection contre les erreurs : Détecter des contraintes contradictoires rendant impossible l'utilisation d'une ou plusieurs actions du problème. Vérifier les saisies en temps réel et signaler les erreurs syntaxiques, lexicales ou sémantiques.

La correction des erreurs : concerne les moyens mis en oeuvres par le système afin de permettre à l'utilisateur de corriger les erreurs. La correction des erreurs est une tâche très délicate qui peut s'avérer très compliquée, surtout quand l'erreur est liée au connaissances du domaine et qu'elle ne soulève pas d'erreurs sémantiques. Par exemple, corriger une erreur conduisant à l'impossibilité de trouver un plan solution est une tâche très difficile. Il est probablement plus important que l'utilisateur comprenne l'échec de produire une solution que de comprendre la solution elle même, et cela pour exploiter au mieux les capacités d'abstraction de l'utilisateur et lui éviter de rejeter le système après plusieurs échecs.

Exemple de correction des erreurs : Signaler que la méthode de réalisation d'une tâche demandée n'est pas définie, ou mettre en avant des contraintes qui ont fait rejeter une solution potentielle. Ou encore, donner des statistiques sur toutes les solutions rejetées.

Qualité des messages : Évalue la gestion par le système des messages d'erreur, leurs type (textuelles, graphiques, coloration en rouge, *etc.*), leurs degré de précision, la nature de l'erreur signalée (syntaxique, sémantique, *etc.*). Les messages d'erreur peuvent signaler des erreurs dans les données du domaine ou du problème, mais aussi, les erreurs lors de l'interaction ou de la planification.

Exemples de message d'erreur : Afficher textuellement ou verbalement une erreur syntaxique ou sémantique dans la définition des données, en affichant la donnée en question ou en pointant vers la ligne ou elle est définie. Signaler un conflit dans les données. Signaler une erreur d'interprétation d'une commande de l'utilisateur. Signaler l'impossibilité de trouver une solution.

La stabilité/Cohérence : S'intéresse aux techniques implémentées dans les systèmes en mixed-initiative afin d'offrir une interaction cohérente et des solutions avec le moins de changements possibles, ce qui est nécessaire à la compréhension des échanges et des solutions par les utilisateurs. Sans considérer les aspects du dialogue et en ne prenant en compte que la planification, deux sous-critères sont nécessaires pour avoir cette homogénéité et cohérence :

Stabilité des solutions : S'intéresse aux techniques mises en place afin d'éviter de créer une confusion chez l'utilisateur en lui proposant des solutions qui lui paressent incohérentes ou éloignées des solutions qu'il a pu avoir précédemment. Dans Ferguson & Allen (2007), la propriété de *modification minimum* ou de *stabilité de plans* entre deux recherches avec des modifications mineures est présentée comme une caractéristique importante que les

systèmes en mixed-initiative doivent gérer afin d'éviter de créer une confusion chez l'utilisateur.

Exemple d'homogénéité de solution : Replanifier à partir de changements apportés par l'utilisateur ne consiste pas seulement à faire une nouvelle recherche à chaque fois que l'utilisateur lance une exécution, mais doit tenir compte des plans générés précédemment, réutiliser le maximum d'entre eux et trouver une solution qui n'est pas nécessairement la meilleure mais qui soit la moins éloignée de la solution initiale.

Comparaison / what-if : La comparaison est essentielle dans un système où la compréhension de la solution est un élément clé de son succès. La comparaison entre deux plans est directement liée à l'homogénéité des solutions. Avec une fonctionnalité de comparaison, l'utilisateur peut comparer deux plans en fonction du nombre d'actions, de performances ou selon une valeur d'objet précise.

Exemple de comparaison : Dans (Allen & Ferguson, 2002), une comparaison est faite entre deux solutions où la première est obtenue en utilisant un hélicoptère et la seconde en utilisant des bus pour transporter des personnes. La comparaison se fait généralement entre deux plans solution obtenus en modifiant une contrainte, un objectif ou l'état du système cible.

La signifiance des dénominations : s'intéresse à la gestion des liens sémantiques entre différents objets définis dans les domaines de planification des systèmes en mixed-initiative.

Sémantique des dénominations : Dans les systèmes permettant l'édition et la modification du domaine par les utilisateurs, plusieurs dénominations et définitions différentes peuvent être données aux mêmes objets apportant ainsi plus d'ambiguïté dans les problèmes. Afin de palier à cela et avoir des définitions homogènes permettant une interaction efficace avec tous les utilisateurs, les systèmes en mixed-initiative doivent gérer cette problématique et fournir les outils nécessaires pour y répondre.

La compatibilité technique : S'intéresse aux caractéristiques techniques des systèmes en mixed-initiative, aux fonctionnalités qu'ils proposent ainsi que leurs compatibilité avec d'autres systèmes ou à une évolution de technologie. Pour cela, les systèmes doivent être analysés avec les critères suivants :

Technique de planification : S'intéresse à la technique de planification automatique utilisée, aux structures des données qu'elle utilise et son degré de compatibilité avec les critères essentiels dans un système de planification en mixed-initiative.

Exemple de technique planification : La planification classique avec en entrée, un problème définit en PDDL et un algorithme de planification en chaînage avant. On peut aussi

avoir un algorithme de planification partiellement ordonné avec des buts ouverts et des menaces à corriger donnant des plans solution partiellement ordonnés.

Interface de communication : S'intéresse aux moyens mis à disposition par le système afin que les différents agents et surtout l'utilisateur et le planificateur puissent communiquer. La prise et le basculement de l'initiative dépend fortement des moyens de communication dont disposent les agents. Ainsi, la prise d'initiative ne se présente pas de la même façon en fonction de l'interface de communication utilisée.

Exemple d'interfaces de communication : un système avec une interface en langage naturel (parlé ou écrit) ; Une interface graphique où l'utilisateur peut cocher, bouger et pointer des icônes ; Une interface gestuelle avec reconnaissance de mouvements (Mistry et al. , 2009; Jiang et al. , 2014).

Représentation des solutions : S'intéresse à la compréhension par l'utilisateur des solutions proposées. Comprendre un plan solution dépend entre autres du degré de son expressivité. On entend par l'expressivité, l'information contenue dans le plan qui peut contenir des actions, des liens de causalité, des préférences, des évaluations, etc. et aussi les possibilités d'interaction offertes par la solution à l'utilisateur, un plan partiellement ordonné, par exemple, offre plus de possibilités.

Exemple de représentation des solutions : Représentation textuelle, totalement ou partiellement ordonnée. Ça peut aussi être un workflow, une suite d'instructions ou un plan offrant la possibilité de déplier et de regrouper des actions, suivre les liens de causalité, etc.

Édition des connaissances : S'intéresse aux capacités du système à fournir les outils et les points d'accès nécessaires à la manipulation des connaissances du domaine par des experts. Il est très compliqué et très onéreux de définir toutes les connaissances d'un domaine donné avant de commencer à exploiter l'outil de planification, d'où l'utilité de permettre l'édition du domaine pour extraire progressivement la connaissance et l'expérience des experts et d'utiliser ces définitions dans des applications futures.

Exemple d'édition du domaine : Les utilisateurs peuvent définir de nouvelles actions, modifier des conditions préalables ou des effets, ou définir une méthode de décomposition pour une tâche si une représentation hiérarchique est utilisée.

3.3 Présentation des systèmes de planification en mixed-initiative

Cette section présente des systèmes fondés sur la planification en mixed-initiative et aborde leurs différentes applications, leurs organisation, leurs fonctionnements ainsi que la prise en charge des fonctionalités de mixed-initiative. Les systèmes présentés sont très hétérogènes et ont été développés pour des applications différentes. Certains sont très dépendants de leurs domaines d'application alors que d'autres peuvent être utilisés dans plusieurs applications. L'analyse de ces systèmes fournit une base pour la comparaison du système développé durant cette thèse et permet d'avoir une meilleure idée sur leurs limites ainsi que sur les points intéressants à retenir pour le développement d'un système de planification en mixed-initiative.

3.3.1 PSM/TRIPS

« Planning Solving Manager » (PSM) (Allen & Ferguson, 2002; Ferguson & Allen, 2007) est un module de planification interactif qui trouve ses racines dans « Trains-95 » (Ferguson et al. , 1996) et qui est ensuite repris dans *« the Rochester Interactive Planner System »* (TRIPS) (Ferguson et al. , 1998) : un système de planification collaboratif indépendant du domaine pour la construction interactive et incrémentale de plans qui a été appliqué aux missions d'évacuation des populations et la planification des transports. Les auteurs définissent quatre besoins de la planification collaborative au quels le système doit répondre :

- Un développement incrémental du plan : considérant une partie du plan avant de passer à la suite.
- Une stabilité dans le plan : Un changement minimum du plan lorsque des contraintes sont ajoutées ou lors d'une re-planification. ce besoin rejoint le critère de stabilité et de cohérence que nous avons défini.
- Une planification avec une stratégie de recherche qui peut changer.
- Un système ouvert à l'innovation, i.e, qui doit être capable de construire sous la direction de l'utilisateur.

Pour répondre à ces besoins, PSM fournit une représentation hiérarchique du plan avec, au niveau supérieur, une représentation hiérarchique des objectifs qui peuvent être divisés en sous-objectifs et capturés lors de l'interaction avec l'utilisateur. Le deuxième niveau, appelé *« plan abstrait »*, présente à l'utilisateur un ensemble de solutions abstraites. Le niveau plan brut fournit une proposition concrète à l'utilisateur d'une séquence d'actions correspondant au plan abstrait. Le niveau inférieur, appelé "le scénario prévu" fournit une représentation de l'environnement et l'état du système lors de l'exécution du plan brut en fonction d'une simulation. Cette décomposition apporte un vrai plus

dans la gestion de la charge de travail et permet de bien répondre au critère de décomposition du problème.

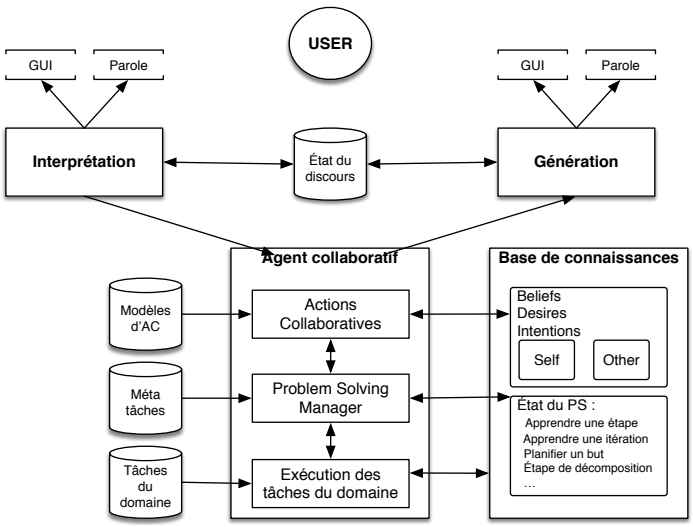


FIGURE 3.4 – Architecture d'un agent collaboratif de dialogue.

Les interactions avec le système sont guidées par les requêtes de l'utilisateur qui peuvent être résumées en trois types d'opérations : (1) La suggestion : l'utilisateur présente un objectif qui n'a pas encore été présenté ; (2) L'extension : l'action proposée étendra le Plan conçu précédemment ; (3) La modification : la proposition d'une action pour remplacer une autre dans un plan existant.

Dans la figure 3.4, on peut voir que le *problem solving manager* récupère les actions collaboratives regroupant les opérations de résolution du problème données par l'utilisateur. Ces actions peuvent être données comme suit : Introduire / Affiner l'objectif, Modifier / corriger l'objectif ou la solution, Évaluer le plan, Spécifier / Étendre la solution, Créer / comparer / rejeter l'option / solution, Annuler l'opération et annuler le plan. Un plan solution est donné après chacune de ces opérations de résolution.

En terme de technique de planification, le système utilise un planificateur à interval temporel logique (Allen & Koomen, 1983; Allen et al. , 1991) similaire à UCPOP (Penberthy et al. , 1992) en supposant des liens de causalité persistants à la place de liens de causalité pour les relations de cause à effet. Les contraintes d'ordre sont déduites des contraintes temporelles et les menaces sont découvertes avec un chaînage arrière en utilisant des axiomes pour détecter l'incohérence temporelle. La décomposition de tâches est effectuée avec un planificateur de type HTN qui décompose les objectifs en sous-tâches avec des contraintes de synchronisation entre elles.

3.3.2 PASSAT

« *Plan-Authoring System based on Sketches, Advice, and Templates* » (PASSAT) (Myers et al. , 2002, 2003) est un framework de planification centré sur l'utilisateur conçu pour des applications dans lesquelles un noyau de connaissances de planification peut être capturé dans des modèles d'action prédéfinis mais où un contrôle significatif des utilisateurs du processus de planification est nécessaire. PASSAT est guidé par deux principes :

- Flexibilité : l'utilisateur peut éditer des modèles pour guider l'exécution des tâches.
- Automatisation ajustable, centrée sur l'utilisateur : la génération du plan est le résultat de la collaboration pas à pas entre le planificateur et l'utilisateur. Cependant, l'automatisation de la procédure de planification peut être décidée par l'utilisateur s'il la juge plus efficace.

PASSAT est fondé sur un modèle HTN classique (Erol et al. , 1994) augmenté avec une représentation temporelle qui permet une spécification d'une décomposition de tâches plus riche que le modèle HTN classique. Pour cela, le système utilise des modèles qui sont des méthodes de décomposition issues du planificateur *HTN UMCP* et représentent la décomposition d'une tâche en sous-tâches. Plusieurs modèles peuvent être définis pour une seule tâche, chacun exprimant une décomposition différente. Les contraintes représentent un prédicat, la représentation des données utilise une logique trivaluée et le système permet à l'utilisateur de relaxer des contraintes lors de la planification interactive. Pour représenter le domaine, le framework utilise une ontologie fondée sur une «Generic Frame Protocol Representation» (Karp et al. , 1995). PASSAT présente deux modes de génération de plan :

- Raffinement interactif du plan : En mode interactif, l'utilisateur peut : (1) décomposer une tâche, en choisissant d'appliquer un modèle, spécifier interactivement un ensemble de sous-tâches, utiliser un sketch ou annuler la tâche. En choisissant un sketch, le système fait correspondre la tâche au modèle, ensuite, il rajoute les sous-tâches et les contraintes du modèle à un agenda qui contient les tâches à décomposer et les contraintes à vérifier ; (2) affecter une valeur à une variable en choisissant une valeur de variable dans une liste qui lui est fournie ou en entrant une nouvelle valeur et justifier ce choix ; (3) relaxer une contrainte : les contraintes qui n'ont pas été vérifiées sont ajoutées à l'agenda et peuvent être relaxées par l'utilisateur dans les cas suivants : Il a de nouvelles informations permettant de valider la contrainte ; il sait que la contrainte est trop forte pour l'état actuel du système ; il veut simplement tester les résultats sans cette contrainte.
- Plan sketching : le planificateur hiérarchique suit un développement descendant, contrairement à l'utilisateur qui tend à suivre un raisonnement ascendant (p. Ex. : Décider d'une action sans connaître la stratégie globale à suivre). Un sketch est une collection de tâches qui peuvent

être partiellement spécifiées à plusieurs niveaux d'abstraction dans la hiérarchie du plan. PASSAT fournit un éditeur interactif pour spécifier un sketch de plan.

3.3.3 HICAP

« *Hierarchical Interactive Case-Based Architecture for Planning* » (HICAP) (Muñoz Avila et al. , 1999; Aha et al. , 2001) est un système d'assistance à la planification des opérations d'évacuation du personnel non-combattant. HICAP intègre un éditeur de décomposition de tâches « *Hierarchical Tasks Editor* » (HTE) qui permet aux utilisateurs de sélectionner et d'éditer des tâches à l'aide d'un planificateur conversationnel NaCoDAE(Giampapa & Sycara, 2001)/HTN offrant la possibilité aux utilisateurs d'affiner les plans de manière interactive. L'utilisateur sélectionne une tâche à décomposer à partir du HTE, puis la raffine de manière interactive avec NaCoDAE/HTN en sélectionnant et en appliquant la décomposition correspondante.

Selon les auteurs, pour être accepté dans un « *Noncombatant Evacuation Operations* » (NEO), un planificateur doit fournir les trois caractéristiques suivantes : (1) être *Doctrine-driven* : Utiliser une analyse des principes de base (doctrine) pour définir des décompositions et guider la planification ; (2) être interactif : l'utilisateur contrôle comment une décomposition abstraite est instanciée ; (3) permettre la sauvegarde des cas : enregistrer des segments des plans précédents et les mettre en sur-brillance pour l'utilisateur s'ils correspondent au cas actuel.

Le *HTE* permet à l'utilisateur d'enlever ou de remplacer une décomposition. Il lui permet également de sélectionner une tâche et sa décomposition à l'aide du décomposeur de tâches conversationnel. En se basant sur son expérience et les ressources disponibles, l'utilisateur peut ainsi adapter une décomposition (doctrine) en ajoutant des tâches et en éliminant celles qui lui paressent non pertinentes. Il peut également utiliser l'expérience acquise par les *NEO* précédents pour mieux affiner le plan en fonction de la situation actuelle.

Le Décomposeur de tâches conversationnel utilise une variante du planificateur *UMCP* (Erol et al. , 1994) et permet de sélectionner de manière interactive une décomposition pour une tâche. Cette dernière ne peut être sélectionnée que si toutes ses contraintes sont satisfaites.

3.3.4 MAPGEN

« *Mixed-Initiative Activity Planning Generator* » (MAPGEN) (Ai-Chang et al. , 2003, 2004; Bresina et al. , 2005; Bresina & Morris, 2007; Smith, 2012) est un système de planification automatique en

mixed-initiative utilisé dans les missions d'exploration spatiale avec des Mars Rover. MAPGEN est utilisé pour gérer les activités des différentes équipes en gérant les priorités et les contraintes des scientifiques, les instances d'activité des instruments définies par les experts, les tâches pour les pilotes des Rovers, les analyseurs de données des Rovers qui donnent les conditions initiales et l'approbation de l'équipe de gestion de la mission. Afin de répondre à une telle charge de travail, MAPGEN combine :

- « *Activity Plan Generator* » (APGEN)(Maldague et al. , 1998) : qui est une interface utilisateur pour un outil de planification multi-mission développé au Jet Propulsion Laboratory et utilisée dans plusieurs missions spatiales. APGEN fournit une interface utilisateur graphique pour visualiser les ressources, créer et modifier manuellement les plans. L'interface d'édition de plan APGEN ne supportant pas l'ajout de contraintes temporelles, un éditeur de contrainte temporelle externe a été ajouté à APGEN pour permettre aux utilisateurs d'afficher les activités et les contraintes existantes, de les ajouter, de les supprimer ou de les modifier. APGEN fournit également une base de données de plan d'activité qui soutient l'expansion de l'activité sans maintenir les contraintes entre elles.
- « *Extensible Universal Remote Operations Planning Architecture* » (EUROPA)(Jonsson et al. , 1999; Frank & Jónsson, 2003; Barreiro et al. , 2012) : qui est une bibliothèque pour le développement de planificateurs avec un paradigme de planification basé sur les contraintes temporelles. Elle propose des outils reposant sur la résolution de problèmes :

  1. Une présentation d'actions, de déclarations, de ressources et de contraintes avec une modélisation orientée objet.
  2. un raisonnement et l'inférence : EUROPA propose des algorithmes d'inférence logique et des traitements de contraintes afin d'appliquer Les règles du domaine
  3. Des heurisiques de recherche : EUROPA fournit également un framework pour intégrer des heuristiques dans les algorithmes de recherche de base ou pour développer de nouveaux algorithmes de recherche.

MAPGEN fournit un processus de planification incrémental en entrelaçant des phases de génération automatiques et d'édition de plans. Cette façon graduelle de construire des plans aide l'utilisateur à mieux comprendre un plan progressivement développé. En outre, le système prend en charge une approche de perturbation minimale en utilisant une heuristique dite de perturbation minimale, qui tente de présenter aux utilisateur des plans avec des modifications minimales lors de l'ajout de plus d'activités à un plan, car les utilisateurs ont tendance à s'attendre à ce que les petites extensions d'un plan ne provoquent que des modifications mineures.

À travers son interface, l'utilisateur peut modifier les plans en ajoutant, en changeant, en dépla-

çant et en effaçant les activités de haut niveau. *EUROPA* gère les activités de bas niveau telles que la navigation et le CPU du rover et fournit aux utilisateurs des services tels que :

- La planification globale : «Planifie tout»
- La planification sélective : «Planifie ceci et tout ce qui s'y rapporte»
- La planification fine des grains : «Planifie ceci et essayez de le mettre ici»

L'utilisateur a aussi la possibilité de mettre les activités dans un "hopper" qui organise celles qui n'ont pas encore été planifiées.

MAPGEN relaxe la complétude pour être plus réactif. Pour y parvenir sans perdre les chances de trouver des solutions, il utilise un algorithme de chaînage arrière qui calcule la difficulté de trouver une solution et si cette activité dépasse une allocation déterminée par sa priorité, l'activité est rejetée du plan.

3.3.5 Système en mixed-initiative pour le guidage d'UAVs

Le système en mixed-initiative pour le guidage des « *Mixed-initiative planning and execution for unmanned aerial vehicles* » (UAV) (Bevacqua et al. , 2015) est un système multimodal pour la recherche et le sauvetage avec de multiples drones en mixed-initiative. Le système fait partie du projet SHERPA (Marconi et al. , 2012). Contrairement à d'autres systèmes de gestion d'UAVs, l'opérateur participe à la recherche et n'est pas dédié à la supervision des robots. L'opérateur supervise les opérations d'un groupe de drones en utilisant : la gestuelle, le dialogue, un joy-pad et une tablette. Les activités des robots sont présentées sous la forme de tâches hiérarchiques instanciées par le système en fonction de l'environnement et des objectifs définis par l'opérateur. Ce dernier interagit en continu avec le système sur différents niveaux d'abstraction, allants de la définition et du changement des tâches de haut niveau jusqu'à l'ajustement des trajectoires des drones sur le terrain.

Les phases de recherche pour les missions de sauvetage sont divisées en 5 étapes que l'opérateur peut ajuster dynamiquement en fonction du contexte. Il peut ensuite définir des objectifs ou donner plus d'informations dans chaque étape à l'aide de primitives prédéfinies. Dans le système, le chemin de recherche est défini par un ensemble de points (x, y). Une zone de recherche est définie par un centre et un rayon. Au cours de l'interaction, une valeur de priorité peut être attribuée à des chemins ou à une zone de recherche, et une méthode d'exploration peut être affectée à chaque zone de recherche ; une valeur de coût est calculée pour les zones et les chemins de recherche en fonction des drones touchés.

Le système permet à l'opérateur de se concentrer sur les activités critiques (inspection visuelle, manœuvre délicate, etc) et de déléguer les tâches de routine (décomposition des tâches, planifica-

tion des chemins, évitement des obstacles, etc) au système autonome de robots. L'interaction avec l'opérateur est intégrée dans la boucle de contrôle du robot pour avoir une interaction naturelle et intuitive. L'architecture du système permet de gérer différents modes de contrôle :

- Autonome : le robot planifie et exécute des tâches complexes sans intervention de l'opérateur.
- Manuel : Le robot est contrôlé à distance par un opérateur.
- Mixed-Initiative : l'utilisateur exécute certaines opérations et le système autonome se configure en fonction des objectifs.

Le « *Multi-Robot Supervisory System* » (MRS) est un module de supervision qui interagit avec un planificateur hiérarchique, affecte les tâches au « *Robot Supervisory System* » (RSS) et contrôle leur exécution. Le MRS doit compléter et affecter des tâches et des squelettes de plans incomplets définis par l'opérateur. Par exemple, l'opérateur spécifie les domaines à explorer sans préciser le drone chargé de le faire. Le MRS récupère les informations nécessaires (position des robots, tâches, sous-tâches en cours d'exécution et niveau de la batterie) et les transmet au MRS qui attribue des tâches à différents drones. Le système doit pouvoir gérer les tâches abstraites et prendre des décisions sans demander sans cesse à l'opérateur des informations ou des tâches à realiser. Afin de gérer la planification et l'exécution en mixed-initiative, le MRS est géré par un système de type « *Belief Desire Intention executive* » (BDI) (Rao & Georgeff, 1991) qui interagit avec le planificateur hiérarchique « *Human Aware Task Planner* » (HATP) (Montreuil et al. , 2007) amélioré avec des capacités de re-planification.

L'architecture du système est modulaire et permet une interaction multimodale entre le robot et l'opérateur. Le module multimodal gère l'interaction avec ses différents aspects. L'interaction avec le système à travers le discours permet de contrôler les mouvements du robot manuellement ou en mixed-initiative. Les mouvements permettent par exemple de pointer une direction. Le joy-pad est utilisé pour contrôler à distance le robot (changer la vitesse, la direction, la hauteur, etc) sans modifier les tâches du robot. La tablette permet d'afficher des informations sur l'environnement et l'état du robot alors que les notifications prioritaires sont transmises par un casque audio.

3.3.6 Gtrans

« *Goal TRANSformation* » (GTrans) (Immaneni & Cox, 2004; Cox & Zhang, 2007) est un système de planification en mixed-initiative qui fournit à l'utilisateur une interface pour le planificateur Prodigy (Veloso et al. , 1995) et qui présente la modélisation de la planification sous forme d'une manipulation de but (Cox & Veloso, 1998). Le système a évolué à partir d'une interface pour le planificateur Prodigy (Cox & Veloso, 1997) vers un système multi-agent appliqué aux situations d'intervention d'urgence où l'utilisateur peut établir et manipuler des objectifs et où il peut également affecter des ressources à un

objectif spécifique. Le mode de recherche est complètement opaque pour l'utilisateur. Une séquence d'utilisation du système commence par la création d'un problème de planification, puis l'invocation du planificateur Prodigy pour trouver une solution. Ensuite, des modifications des buts suivies de re-planifications sont effectuées jusqu'à ce qu'une solution soit trouvée et validée par l'utilisateur. *GTrans* fournit trois façons de transformer le but :

— La transformation du type de but : Remplacer le prédicat du but par une hiérarchie d'abstractions définie dans le domaine.
— La transformation de l'argument du but : en le remplaçant par une hiérarchie d'abstraction.
— La transformation de la valence : en changeant la valeur de vérité d'un fait du but.

Lors d'une exécution, l'utilisateur définit le problème via l'interface GTrans en définissant les objets, en précisant l'état initial et l'état de but. Prodigy/Agent (intermédiaire entre GTrans et Prodigy) lance le planificateur avec les paramètres entrés par l'utilisateur et renvoie le résultat si un plan est trouvé ou une erreur sinon. À travers les mécanismes de transformation de buts, GTrans permet d'obtenir des plans partiels.

3.3.7 CMMD

« *Coordinated Multi-source Maintenance on Demand* » (CMMD) (Kichkaylo et al. , 2007) est un système d'aide à la décision en mixed-initiative destiné à la planification des missions habitées de la NASA. Ces missions sont réalisées en suivant des procédures et des règles testées et très strictes, mais la planification des missions est extrêmement dynamique et nécessite parfois la levée de certaines restrictions. Le plan est raffiné progressivement jusqu'à ce que les calendriers quotidiens de tous les intervenants soient générés. La plus grande unité de temps considérée pour l'« *International Space Station* » (ISS) est l'*«incremento»* qui correspond à la période que passe une équipe à bord de la station (6 mois). La planification commence 18 mois avant le départ compte tenu de la rotation des équipes, de la formation, etc. Le processus de raffinement n'est pas nécessairement TOP-DOWN, des objectifs de haut niveau peuvent être ajoutés, des plans modifiés, supprimés et détaillés montrent ce qui est réalisable.

Le système s'intéresse beaucoup à l'exploration des solutions alternatives lors de la génération de plans. En partant du principe que les planificateurs modernes peuvent être utilisés pour les applications du monde réel, mais que l'acceptabilité des solutions générées automatiquement ne sont pas faciles à comprendre par les utilisateurs qui doivent les valider et les appliquer. Une explication postérieure n'est pas suffisante lorsque la solution est très complexe. L'objectif de *CMMD* est alors d'être ; Simple ; Interactif ; compréhensible et contrôlable.

Le système possède une base de connaissance contenant des copies des plans applicables nécessitants une propagation des contraintes lors d'une inférence. Une interface utilisateur permet d'interagir avec les utlisateurs et leur permet notamment d'ajouter des informations, le choix des ressources, de suivre la propagation des contraintes, d'évaluer des alternatives, etc. L'utilisateur a la possibilité de demander une solution automatique ou de participer à la recherche pour guider le processus de recherche.

CMMD possède une représentation du domaine similaire aux HTN (Erol et al. , 1994). Il implémente deux types de règles : les « *achievement rules* » et les « *safety rules* ». Les « *achievement rules* » correspondent aux décompositions HTN traditionnelles et précisent un moyen possible d'atteindre un objectif. Les « *safety rules* » prescrivent des conditions supplémentaires, nécessaires pour le déroulement de la mission en toute sécurité et correspondent aux contraintes de sécurité ou aux règles de vol.

3.3.8 Autres systèmes

Il existe dans la littérature d'autres planificateurs en mixed-initiative qui ont été développés pour des applications spécifiques tels que :

MixPlan (Dagkli & Refanidis, 2009) : un outil pédagogique destiné aux étudiants pour les aider à résoudre des problèmes de planification temporelle en leurs permettant de suivre la génération d'un plan, de choisir la direction de la recherche à chaque fois que c'est possible, de revenir à n'importe quel choix effectué auparavant et automatiquement backtracker en cas d'échec. MixPlan implémente l'algorithme « *Partial Order Plan* » (POP) sur des domaines temporels et permet donc une représentation partielle du plan sous forme d'un graphe ordonné. Le système fournit aussi une représentation graphique des plans partiels sous forme de graphe orienté.

« Mixed-InitiativeCase-Based Planning system » **(MI-CBP)** (Veloso et al. , 1997), est le résultat de l'intégration de deux systèmes : ForMAT (Force Management and Analysis Tool) (Mulvehill, 1996) : un planificateur «*case based*» guidé par l'utilisateur qui ne gère ni la construction automatique des plans ni l'adaptation des plans existants. Et le planificateur Prodigy (Veloso et al. , 1995) qui génère des plans automatiquement, les interprète et les sauvegarde pour être réutilisés s'ils sont similaires au nouveau problème. MI-CBP permet de présenter des suggestions pour les modifications qui sont présentées à l'utilisateur via une interface graphique. Il incorpore aussi un éditeur de but et des mécanismes pour la vérification de leur cohérence.

Conclusion

Nous avons, dans ce chapitre, défini de façon formelle les notions d'interaction et de planification en mixed-initiative et avons proposé une grille d'évaluation des systèmes basés sur cette technique suivie d'un état de l'art dans ce domaine. Nous avons vu que la planification en mixed-initiative était largement utilisée dans les systèmes d'assistance et permettait de bien répondre aux attentes de ces derniers.Nous avons vu aussi, à travers les différentes fonctionalités implémentées par ces systèmes ainsi que leurs représentations des connaissances et l'interaction avec l'utilisateur que ces systemes répondent à certains des critères qui ont été précédemment posés. Cependant différents verrous liés à la représentation des connaissances et à l'expressivité des techniques de planification empêchent ces systèmes de répondre de façon optimale à tous les critères d'interaction qui ont été posés et au besoins de utilisateur.

Deuxième partie

Un système de planification en mixed-initiative fondé sur une approche de planification HTN totalement instanciée

4

GTOHP : Un planificateur HTN avec des problèmes instanciés et simplifiés

Sommaire

4.1 La complexité de l'instanciation des problèmes HTN 50

4.2 Algorithme d'instanciation et de simplification des problèmes HTN 52

4.3 Algorithme de planification HTN avec des problèmes instanciés et simplifiés 61

4.4 Expérimentations de l'algorithme d'instanciation et de simplification 62

Introduction

Nous allons, dans cette deuxième partie, aborder les contributions scientifiques et techniques de cette thèse. Pour cela nous allons présenter dans ce chapitre un algorithme d'instanciation et de simplification des domaines et des problèmes HTN, publié dans (Ramoul et al. , 2016), permettant de réduire leur complexité ainsi que celle de la procédure de planification. Nous aborderons, ensuite, l'algorithme du planificateur GTOHP, un planificateur HTN à espace d'états qui raisonne à partir de données instanciées et simplifiés afin de trouver des solutions de façon plus rapide et plus facilement compréhensible par l'utilisateur. Ces deux contributions permettent de répondre aux exigences de la planification en mixed-initiative et servent de base au développement du système de planification *MIP* que nous présentons dans le chapitre 5. Pour finir, nous évaluons les apports d'une étape d'instanciation et de simplification à travers l'étude des performances du planificateur *GTOHP* par rapport à un planificateur *HTN* équivalent.

4.1 La complexité de l'instanciation des problèmes HTN

Le processus d'instanciation consiste à générer toutes les instances possibles des opérateurs, des méthodes et des prédicats à partir des objets définis dans un domaine et dans un problème de planification. Une instance d'un opérateur est appelée *action*, celle d'une méthode est appelée *décomposition*, et celle d'un prédicat est appelée *fait*. Pour générer toutes les instances possibles, tous les paramètres typés dans les opérateurs, les méthodes et les prédicats sont remplacés par les objets correspondants [1]. Le nombre d'instances et la complexité du processus d'instanciation dépendent fortement du nombre de paramètres définis dans les méthodes et les opérateurs ainsi que du nombre d'objets qui sont définis pour chaque type

Si x_i est un paramètre et $D(x_i)$ son domaine, i.e. l'ensemble de tous les objets possibles qui peuvent être associées à ce paramètre. On note $|D(x_i)|$ qui est la cardinalité de $D(x_i)$. Le nombre d'instances $I(o)$ d'un opérateur o avec n paramètres $\{x_1, ..., x_n\}$ ayants comme domaine $D(x_i)$ est :

$$I(o) = \prod_{i=1}^{n} |D(x_i)|$$

On voit que le nombre d'instances générées augmente rapidement avec le nombre de paramètres

1. On dit qu'un objet correspond à une variable s'il possède le même type ou que son type est un sous-type de celui de la variable

dans l'opérateur et la cardinalité de leurs domaines, ce qui peut conduire à une explosion de la complexité du processus d'instanciation et du coefficient de branchement de l'espace de recherche. Par exemple, si l'on considère l'instanciation et la simplification du domaine *Rover* avec le problème *40* issues de la compétition internationale de planification automatique *IPC-5*, le nombre d'actions résultant de l'instanciation et simplification de l'opérateur *communicate_soil_data (?x - rover?l - lander?p1 - waypoint?p2 - waypoint?p3 - waypoint)* est égal à *14 millions* = $(14 \times rover) \times (1 \times lander) \times (100 \times waypoint1) \times (100 \times waypoint2) \times (100 \times waypoint3)$.

La complexité du processus d'instanciation des problèmes HTN est sensiblement plus élevée, puisqu'elle consiste en un processus d'instanciation des méthodes en plus de celui des opérateurs. En sachant que les tâches primitives définies dans les méthodes correspondent aux opérateurs, le nombre de paramètres dans ces méthodes est alors égal à la somme des paramètres des tâches primitives sans répétition. Par conséquent, la complexité de l'instanciation des méthodes est plus élevée ou au moins égale à celle des opérateurs. Cependant, la différence entre le nombre d'instances des méthodes et le nombre d'instances des opérateurs dépend de la définition du problème de planification.

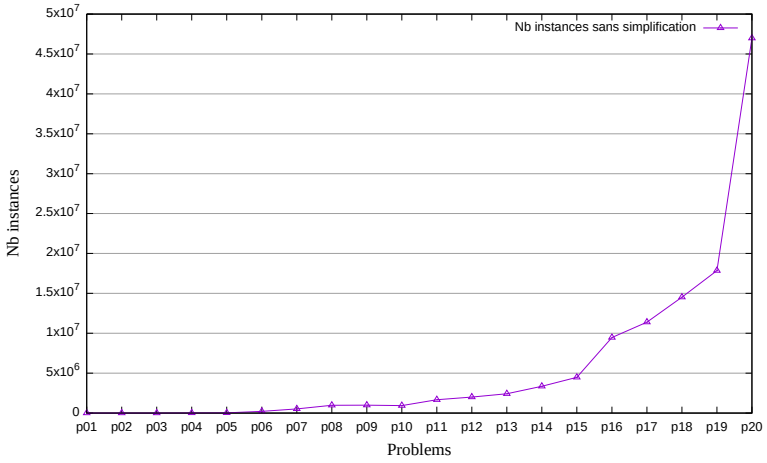


FIGURE 4.1 – Nombre d'instances de méthodes du domaine *Rover* sans simplification

La figure 4.1 présente le nombre de décompositions possibles générées après l'instanciation sans simplification de 20 problèmes du domaine *Rover*, triés par ordre croissant de leur nombre d'objets. On peut voir qu'avec le même nombre de méthodes dans la définition du domaine, le nombre d'instances des méthodes augmente très rapidement avec le nombre croissant d'objets des problèmes. Prenons comme point de comparaison les problèmes 1, 10 et 20 afin d'analyser le lien entre le nombre

d'objets et le nombre de méthodes instanciées. On a 14 objets dans le problème 1, 71 dans le problème 10 et 158 dans le problème 20. Le nombre d'objets du problème 10 au problème 20 a été multiplié par 2,22 alors que celui des décompositions a été multiplié par 50,08. Le nombre d'instances de méthodes augmente beaucoup plus rapidement que le nombre d'objets dans les problèmes. L'augmentation de la complexité accentue l'importance de l'efficacité des mécanismes de simplification lors de l'instanciation, surtout quand on sait que le nombre de décompositions influence fortement le facteur de branchement de l'espace exploré par l'algorithme de décomposition HTN et par conséquent les performances de la planification.

4.2 Algorithme d'instanciation et de simplification des problèmes HTN

Le processus d'instanciation et de simplification, appelé aussi processus de *grounding*, consiste à réaliser plusieurs opérations sur les domaines et les problèmes HTN, afin de calculer toutes les instances des opérateurs et des méthodes qui sont pertinentes dans le problème en question. Ce processus est décrit dans la figure 4.2. A partir d'un domaine et d'un problème HTN décrit en utilisant le langage *HPDDL* (cf section. 2.5), nous nous appuyons sur le processus présenté dans (Koehler & Hoffmann, 1999) pour produire une table de faits spécifiques appelés *inerties* (cf section. 4.2.1). En utilisant cette table d'inertie, le processus d'instanciation et de simplification des opérateurs (section 4.2.2) génère un problème semi-instancié contenant des faits et des actions pertinentes[2]. À La fin du processus d'instanciation et de simplification des méthodes (section 4.2.3), le problème et le domaine HTN sont totalement mis à plat et ne contiennent que des faits, des actions et des décompositions essentielles au processus de planification sans perte de connaissance. Ce problème instancié et simplifié est aussi appelé *grounded-problem*.

4.2.1 Calcul des inerties

Le concept d'inertie, définit dans (Koehler & Hoffmann, 1999), représente des faits qui ne sont jamais produits ou consommés par les actions d'un problème. Il y a deux types d'inertie :

Une inertie positive représente les faits qui n'apparaissent jamais dans les effets positifs de toutes les actions du problème. Cela signifie que ce fait ne peut jamais être produit lors de l'exécution des actions du plan. Par conséquent, si un fait est une inertie positive et n'est pas vrai dans l'état

2. Un fait, une action ou une décomposition sont dit pertinents dans un problème de planification si leur suppression peut affecter le processus ou le résultat de la planification de ce problème

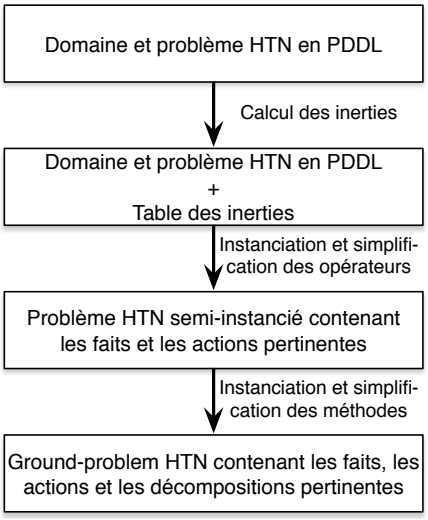


FIGURE 4.2 – Vue globale de l'algorithme d'instanciation et de simplification des problèmes *HTN*.

initial du problème, il ne sera jamais vrai dans aucun état de ce problème.

Une inertie négative représente les faits qui n'apparaissent jamais dans les effets négatifs de toutes les actions du problème. Cela signifie que ce fait ne peut pas être consommé pendant l'exécution des actions du plan. Si un fait est une inertie négative et est vrai dans l'état initial du problème, il sera toujours vrai dans n'importe quel état de ce problème.

Si un fait est une inertie positive et négative en même temps, on dit que c'est une « *inertie* ». Un fait est dit « *fluent* ») s'il n'est ni une inertie positive, ni une inertie négative et qu'il peut apparaître ou disparaître d'un état à un autre. Définir si tous les faits du problème sont des inerties ou non peut être fait en un seul passage sur l'ensemble des opérateurs. En pratique, le calcul de l'inertie se fait en deux étapes au cours du processus d'instanciation des opérateurs, comme expliqué dans Koehler & Hoffmann (1999). La première étape est réalisée avant l'instanciation des opérateurs, avec des prédicats partiellement instanciés et simplifiés, afin de gérer la complexité de l'instanciation et réduire le nombre des actions et des décompositions générées tout au long du processus. La seconde étape est effectuée après l'instanciation, afin de simplifier complètement les actions et décompositions qui ont été produites, en enlevant celles qui ne sont pas pertinentes.

Le tableau 4.1 montre des exemples d'inertie dans le domaine *Rover*. Les prédicats *can_traverse*, *visible* et *at_lander* n'apparaissent ni dans les effets positifs ni dans les effets négatifs des opérateurs du domaine *Rover*, ce qui en fait des inerties. Les inertie positives *at_soil_sample* et *at_rock_sample* n'apparaissent dans aucun effet positif des opérateurs, donc ils ne seront jamais produits. *at* et *visited*

sont des prédicats courants, donc ils peuvent être ajoutés ou supprimés durant la transition inter-états.

Prédicat	Inertie positive	Inertie négative	Courant
available	Non	Oui	Inertie négative
at	Non	Non	Courant
visible	Oui	Oui	Inertie
can_traverse	Oui	Oui	Inertie
communicated_soil_data	Non	Oui	Inertie négative
communicated_rock_data	Non	Oui	Inertie négative
at_soil_sample	Oui	Non	Inertie positive
at_rock_sample	Oui	Non	Inertie positive
at_lander	Oui	Oui	Inertie
visited	Non	Non	Courant

TABLE 4.1 – Exemple de table d'inertie du domaine *Rover*

4.2.2 Instanciation et simplification des opérateurs

Le processus d'instanciation et de simplification des opérateurs génère uniquement les actions pertinentes du problème. Il suit les quatre étapes représentées dans la figure 4.3. À partir de l'ensemble des opérateurs définis dans le domaine et de la table d'inerties calculées dans l'étape précédente, il normalise et instancie les opérateurs lors des étapes de (1) normalisation et (2) d'instanciation et de simplification des opérateurs. Après cela, les formules atomiques dans les opérateurs sont simplifiées (3) en utilisant la table d'inerties et des règles de tautologies des formules logiques du premier ordre. enfin, l'ensemble des actions simplifiées est produit en supprimant celles qui ne sont pas pertinentes, en suivant les règles de simplification (4).

Étape 1. (Normalisation des expressions logiques des opérateurs) Pour chaque expression logique définie dans les pré-conditions et les effets d'un opérateur et contenant des implications ou des quantificateurs, une re-formulation sous forme normale conjonctive est une condition préalable à toute manipulation. Pour réaliser cette opération, nous effectuons les transformations logiques suivantes :

— $\phi \rightarrow \varphi \Rightarrow \neg\phi \wedge \varphi$

— $\neg(\phi \wedge \varphi) \Rightarrow \neg\phi \vee \neg\varphi$

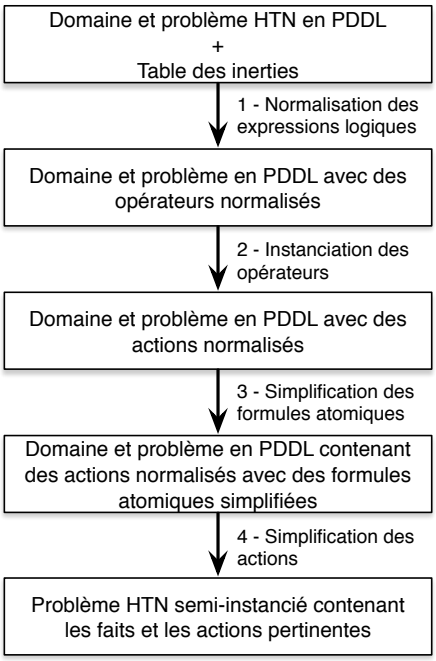


FIGURE 4.3 – Vue globale de l'algorithme d'instanciation et de simplification des opérateurs.

— $\forall(?x - type) \Rightarrow x_1 \wedge x_2 \wedge ... \wedge x_n$

— $\exists(?x - type) \Rightarrow x_1 \vee x_2 \vee ... \vee x_n$

Étape 2. (Instanciation des opérateurs) Comme expliqué dans la section 4.1, l'instanciation des opérateurs consiste à générer toutes les actions possibles en remplaçant les variables par toutes les combinaisons d'objets. L'action *navigate* de l'exemple suivant est une instance possible de l'opérateur *(navigate)* présenté dans le chapitre 2.

```
(:action navigate
 :parameters (rover w1 w0)
 :precondition (and (available rover) (can_traverse rover w1 w0)
                    (visible w1 w0) (at rover w1))
 :effect (and (not(at rover w1)) (at rover w0)))
```

Les paramètres ont été instanciées comme suit : $?x \rightarrow rover$, $?from \rightarrow w1$, $?to \rightarrow w0$. Le processus d'instanciation et de simplification boucle sur la liste des opérateurs et définit les objets correspondants pour chaque variable en fonction de leur type. Si le problème n'est pas typé [3], il est nécessaire d'effectuer une étape d'inférence de type avant l'instanciation. Le processus est

3. Un problème non typé est un problème qui ne définit pas de type de variable

expliqué en détail dans (Koehler & Hoffmann, 1999).

Étape 3. (Simplification des formules atomiques) Dans un souci de performance et dans le but de réduire le coût du processus d'instanciation et de simplification, la simplification des formules atomiques doit être effectuée le plus tôt possible durant le processus d'instanciation. En utilisant la table des inerties, calculée précédemment, la simplification des formules atomiques consiste à évaluer les formules logiques contenues dans les pré-conditions et les effets des opérateurs à *vrai* ou *faux*. Si p est un prédicat contenu dans une formule atomique d'un opérateur et s_0 l'état initial, la simplification des formules atomiques des opérateurs est effectuée en suivant les règles suivantes :

— **Si** p est une inertie positive et $p \notin s_0$ **alors** p est simplifié à **faux**.

— **Si** p est une inertie négative et $p \in s_0$ **alors** p est simplifié à **vrai**.

— **Sinon** p ne peut pas être simplifié.

Dire que $p \notin s_0$ est une simplification, cela signifie que tous les faits issus de l'instanciation et de la simplification de p sont *vrais* dans l'état initial s_0. Lorsqu'un prédicat est évalué à *faux*, cela signifie que tous les fais correspondants ne serons jamais *vrais* dans tous les états possibles du problème. De même, si un prédicat est évalué à *vrais*, tous les faits correspondants seront toujours *vrai* dans tous les états possibles du problème.

Si on considère, par exemple, la table d'inerties du domaine *Rover*, où *(can_traverse)* est une inertie positive et négative, et en supposant que le fait *(can_traverse rover w1 w0)* est vrai dans l'état initial, la simplification le remplacera par *vrai* dans toutes les actions. Par exemple l'action *(navigate rover w1 w0)* sera simplifiée comme suit :

```
(:action navigate
 :parameters (rover w1 w0)
 :precondition (and (available rover) (true)
                    (visible w1 w0) (at rover w1))
 :effect (and (not(at rover w1)) (at rover w0)))
```

Supposons, maintenant, que *(can_traverse rover w1 w0)* soit *faux* dans l'état initial, alors il sera remplacé par *faux* dans toutes les actions. Dans le cas de *(navigate rover w1 w0)*, toutes les pré-conditions de l'action seront évaluées à *faux* comme montré dans l'exemple suivant :

```
 :precondition (and (available rover) (false)
                    (visible w1 w0) (at rover w1))  =>  :precondition (false)
```

Étape 4. (Simplification des actions) Le but est de trouver et de supprimer des actions qui ne pourront jamais être appliquées dans le problème en se basant sur le résultat de la simplification des formules atomiques présentées précédemment. Comme on a pu le voir, les expressions atomiques peuvent être simplifiées à *vrai* ou à *faux* et impliquer la simplification des préconditions et des effets en appliquant des règles de transformation logique. Si ces simplifications permettent de réduire à *vrai* ou *faux* toute l'expression logique contenue dans les préconditions ou les effets d'une action, cette dernière peut être simplifiée comme suit :

— Si la pré-condition d'une action est remplacée par *faux*, l'action est supprimée du problème de planification : Si la pré-condition est fausse, cela veut dire qu'elle nécessite un fait qui ne peut jamais être *vrai* dans le problème, l'action ne peut, donc, jamais être appliquée dans le problème.

— Si tous les effets de l'action sont évalués à *vrai*, l'action ne produit que des faits qui sont toujours *vrais* ou ne consomme que des faits qui sont toujours *faux*. Elle peut, donc, être supprimée du problème car elle ne produit aucun changement.

4.2.3 Instanciation et simplification des méthodes

Le processus d'instanciation et de simplification des méthodes suit les cinq étapes données dans la figure 4.4 ; (1) les expressions logiques contenues dans les contraintes des méthodes sont normalisées ; (2) les types des variables non déclarés dans les paramètres des méthodes sont inférés ; (3) les méthodes sont instanciées ; (4) les formules atomiques contenues dans les contraintes des méthodes sont simplifiées en se fondant sur la simplification des formules atomiques, présentées précédemment ; (5) les décompositions non pertinentes, i.e., les méthodes dont des tâches correspondant à des opérateurs qui ont été supprimés dans les simplifications précédentes, ou dont des contraintes ont été simplifiées à *faux*, sont supprimées du problème. Le résultat final de l'instanciation des méthodes est un problème d'instanciation ne contenant que des faits, des actions, et des décompositions pertinentes.

Étape 1. (Normalisation des expressions logiques) La normalisation des expressions logiques transforme toute les expressions logiques contenues dans les contraintes des méthodes en une forme conjonctive normale nécessaire aux futures manipulations (se référer à la section 4.2.2 pour plus de détails)

Étape 2. (Inférence des types des paramètres) Dans les méthodes, les variables peuvent être utilisées dans des sous-tâches et des contraintes sans être déclarées dans les paramètres des méthodes. Ces variables n'ont pas de type définit explicitement, ce qui nécessite de déduire leurs

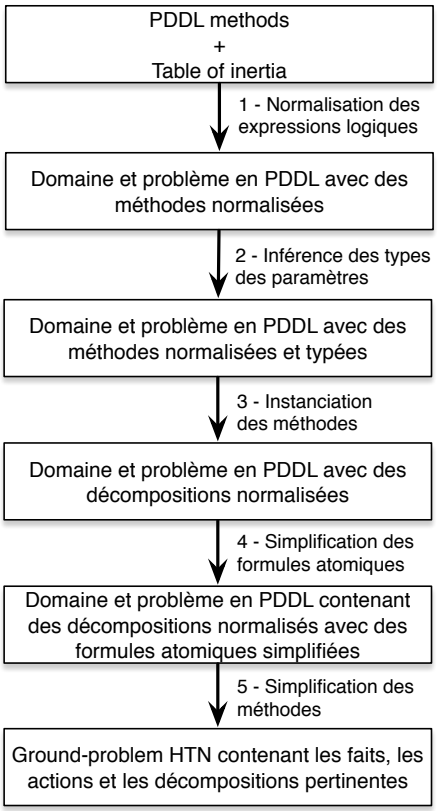


FIGURE 4.4 – Vue globale de l'algorithme d'instanciation et de simplification des méthodes.

types avant de pouvoir effectuer l'instanciation et la simplification des méthodes. Le processus d'inférence de type pour chaque variable non déclarée se fait en deux phases :

Phase 1. Inférence des types à partir des sous-tâches :

- Récupérer T, l'ensemble des sous-tâches contenant dans leurs paramètres la variable non déclarée.
- Pour chaque sous-tâche $t \in T$, récupérer l'opérateur op ou les méthodes M non définies pour t.
- Pour chaque sous-tâche $t \in T$, récupérer les types déclarés dans les paramètres de op ou m. Si le résultat comporte deux types A et B, où B est un sous-type de A, alors garder le type B [4].
- Si le résultat comporte plusieurs types n'ayant aucun lien d'héritage entre eux, une erreur de typage est signalée.

Phase 2. Inférence des types à partir des contraintes :

4. On garde toujours le type le plus spécifique parce c'est celui qui contraint le plus les objets correspondant à la variable . Tous les objets du type B sont aussi du type A, mais l'inverse n'est pas vrai.

— Récupérer P, l'ensemble des prédicats utilisés dans les contraintes de la méthode et contenant la variable non déclarée dans leurs paramètres.
— Pour chaque prédicat $p \in P$, récupérer les types de variables non déclarées à partir de la définition des prédicats dans le domaine.
— Si le résultat comporte plusieurs types n'ayant aucun de lien d'héritage, une erreur de typage est signalée. Sinon,les types ont un lien d'héritage, garder le plus générique.

Étape 3. (Instanciation des méthodes) Comme dans l'instanciation des opérateurs, l'instanciation d'une méthode remplace tous ses paramètres déclarés et non-déclarés par les objets correspondants. Par exemple, la décomposition *do_navigate-rover-w1-w0* présentée ci-dessous est une instance de la méthode *(do_navigate ?r - ROVER ?from - WAYPOINT ?to - WAYPOINT)* introduite dans l'exemple 2.6.1, avec les associations suivantes : $?x \rightarrow rover$, $?from \rightarrow w1$, $?to \rightarrow w0$, $?mid \rightarrow w3$. la valeur $w3$ affectée à $?mid$ correspond au type *WAYPOINT*, en se basant sur le résultat de l'inférence des types. :

```
(:method do_navigate
 :parameters  (rover w1 w0)
 :expansion   ((tag t1 (navigate rover w1 w3))
              (tag t2 (visit w3))
              (tag t3 (do_navigate rover w3 w0))
              (tag t4 (unvisit w3)))
 :constraints (and
                (before (and (not(can_traverse rover w1 w0)) (not(visited w3))
                             (can_traverse rover w1 w3)) t1)))
```

Étape 4. (Simplification des formules atomiques) Cette simplification vise à évaluer les formules atomiques contenues dans les contraintes des méthodes à *vrai* ou à *faux*, en utilisant le concept d'inertie. Elle est fondée sur la procédure présentée pour les opérateurs. L'exemple qui suit montre la simplification de la formule atomique contenue dans la contrainte *before* définie dans la méthode *do_navigate*.

```
(:method do_navigate
 :parameters  (rover w1 w0)
 :expansion   ((tag t1 (navigate rover w1 w3))
              (tag t2 (visit w3))
              (tag t3 (do_navigate rover w3 w0))
              (tag t4 (unvisit w3)))
 :constraints (and
```

```
(before (and (not(false)) (not(visited w3))
             (can_traverse rover w1 w3)) t1)))  =>  (before (false) t1)
```

En sachant que le prédicat *(can_traverse)* est une inertie dans le domaine *Rover*, et dans l'hypothèse que le fait *(can_traverse rover1 waypoint3 waypoint0)* est *vrai* dans l'état initial, il sera simplifié à *vrai* et toute l'expression logique de la contrainte *before* sera simplifiée à *faux*.

Étape 5. (Simplification des méthodes) La simplification des méthodes vise à identifier et supprimer les méthodes contenant des contraintes qui ne peuvent jamais être vérifiées dans le problème. Deux types de simplification sont réalisés sur les méthodes :

Simplification fondée sur les contraintes. Elle repose sur les évaluations des expressions logiques, qui peuvent être simplifiées à *vrai* ou *faux* durant l'étape précédente. La simplification suit les règles suivantes :

— Si l'expression logique est simplifiée à *vrai*, la contrainte est supprimée, car elle est toujours vérifiée.

— Si l'expression logique est simplifiée à *faux*, la méthode entière est supprimée. Dans ce cas, cette contrainte ne peut jamais être vérifiée dans le problème, ce qui signifie qu'elle ne pourra jamais être appliquée ou ne conduira jamais à un plan solution.

Simplification fondée sur les tâches. Elle vise à supprimer les décompositions contenant des tâches primitives qui ne peuvent pas être appliquées. En sachant que la simplification des opérateurs est réalisée avant celle des méthodes, la procédure de simplification est effectuée comme suit :

1. Récupérer T, l'ensemble des tâches primitives définies dans la décomposition à simplifier
2. Pour chaque tâche $t \in T$, vérifier si l'action pertinente pour t a été supprimée lors de la phase de simplification des opérateurs. Si c'est le cas, alors la décomposition entière est supprimée.

Une autre simplification des méthodes fondées sur les tâches composées pourrait être réalisée. Dans cette simplification, une décomposition est supprimée si elle contient une tâche composée qui n'a pas de méthode pertinente. Cela se produit si toutes les décompositions pertinentes pour une tâche sont supprimées par le processus de simplification effectué auparavant. Cette simplification nécessiterait de nombreuses itérations car chacune d'entre elles pourrait simplifier toutes les méthodes pertinentes pour une tâche et impliquer une nouvelle itération. Le processus se poursuivrait jusqu'à la stabilisation de l'ensemble des décompositions pertinentes.

4.3 Algorithme de planification HTN avec des problèmes instanciés et simplifiés

L'algorithme 4.1 montre la procédure *GTOHP* générique et non-déterministe pour résoudre un problème de planification HTN totalement instancié et simplifié. L'algorithme est fondé directement sur la construction récursive d'un plan solution pour le problème de planification HTN. *GTOHP* est un algorithme de planification totalement ordonné, donc, il ne nécessite que des pré-conditions pour vérifier l'applicabilité des décompositions dans l'état courant. Le langage de définition des méthodes et l'algorithme d'instanciation et de simplification que nous avons proposé répondent parfaitement à cette exigence, en considérant l'ensemble des contraintes *before* d'une décomposition comme des pré-conditions à son application.

Algorithme 4.1 : GTOHP(s_0, T, A, D)

Let $T = \langle t_1, \ldots, t_k \rangle$

if $k = 0$ **then** retourner le plan vide $\langle\rangle$

if t_1 est primitive **then**

- $A_{pertinentes} \leftarrow \{a \mid a \in A$ tel que a est pertinente pour t_1 et a est applicable dans $s_0\}$
- **if** $A_{pertinentes} = \emptyset$ **then** retourner échec
- Choisir de manière non déterministe une action $a \in A_{pertinentes}$
- $\pi \leftarrow GTOHP(\gamma(s_0, a), \langle t_2, \ldots, t_k \rangle, A, D)$
- **if** $\pi =$ **échec then** retourner **échec**
- **else** retourner $a \oplus \pi$

else if t_1 est non-primitive **then**

- $D_{pertinentes} \leftarrow \{d \mid d \in D$ tel que d est pertinente pour t_1 et d est applicable dans $s_0\}$
- **if** $D_{pertinentes} = \emptyset$ **then** returner échec
- Choisir de manière non déterministe une décomposition $d \in D_{pertinentes}$
- retourner GTOHP($s_0, subtasks(d) \oplus \langle t_2, \ldots, t_k \rangle, A, D$)

La procédure HTN générique prend en entrée un problème $P = (s_0, T, A, D)$ ou s_0 est l'état initial, $T = \langle t_1, t_2, ..., t_k \rangle$ est une liste de taches, A, l'ensemble des actions, D, l'ensemble des décompositions. Premièrement, l'algorithme teste si la liste des tâches T est vide (ligne 2). Dans ce cas, aucune tâche ne doit être exécutée, donc le plan vide est retourné. Ensuite, la procédure obtient la première tâche t_1 de la liste T. Deux cas doivent être considérés en fonction du type de t_1 :

Cas 1. Si t_1 est primitive (ligne 3) alors l'algorithme récupère, depuis l'ensemble des actions pertinentes du problème instancié et simplifié, l'ensemble des actions qui accomplissent t_1 et qui sont applicables dans s_0 (ligne 4). Cet ensemble peut contenir, au maximum, une action, puisque, dans notre modèle, à chaque tâche primitive correspond une seule action. S'il n'y a pas d'action (ligne 5), la procédure échoue car aucune action ne permet de réaliser la tâche t_1 dans l'état s_0. Sinon ; la procédure choisit de manière non déterministe une action qui accomplit la tâche (ligne 6) et s'appelle récursivement avec le problème de planification $P' = (\gamma(s_0, a_1), T - \{t_1\}, A, D)$ (ligne 7). Enfin, si l'appel récursif de la procédure ne parvient pas à trouver un plan π, il renvoie *« échec »* (ligne 8) ; sinon, il retourne le plan qui est la concaténation de a et π (ligne 9).

Cas 2. Si t_1 est non-primitive (ligne 10) alors l'algorithme récupère, depuis l'ensemble des décompositions pertinentes du problème instancié et simplifié, l'ensemble des décompositions qui décomposent t_1 et qui sont applicables dans s_0 (ligne 11). S'il n'y a pas de décomposition à accomplir t_1 (ligne 12) alors la procédure retourne *échec*. Ensuite, l'algorithme choisit de façon non déterministe une décomposition d qui accomplit la tâche t_1 (ligne 13) et renvoie récursivement le plan solution pour le problème $P' = (s_0, subtasks(d) \oplus \langle t_2, \ldots, t_k \rangle, A, D)$ (line 14).

GTOHP maintient un état tout au long de la procédure de construction de plan solution, et décompose les tâches du task network selon leurs ordre de définition, produisant un plan totalement ordonné. Cependant, le langage de définition *HPDDL* et l'algorithme d'instanciation et de simplification permettent de générer des problèmes instanciés et simplifiés utilisables pour des méthodes de planification différentes, allant de la planification classique jusqu'à la planification HTN partiellement ordonnée, en passant par la planification HTN totalement ordonnée. Offrant une solution à la problématique de test et de comparaison d'algorithmes de planification, en répondant au problème d'hétérogénéité des langages de définition et d'implémentation. Un autre planificateur *HTN* partiellement ordonné, fondé sur l'algorithme *UMCP* (Erol et al. , 1994) est en cours de développement. Cet algorithme prend en entrée un problème instancié et simplifié et utilise les calculs d'atteignabilité dans le but d'améliorer les performances des planificateurs *HTN* partiellement ordonnés.

4.4 Expérimentations de l'algorithme d'instanciation et de simplification

Dans cette deuxième partie de chapitre, nous présentons les résultats obtenus lors de l'évaluation de l'algorithme d'instanciation et de simplification. Nous abordons les performances du planificateur *GTOHP* et nous les comparons à ceux d'un autre planificateur implémentant la version originale de l'algorithme *SHOP* (Nau et al. , 1999), afin de quantifier l'amélioration des performances apportées

par le processus d'instanciation et de simplification pour ce type d'algorithmes HTN.

4.4.1 Cadre expérimental

Afin d'évaluer les performances du processus d'instanciation et de simplification, une série de tests a été réalisée et développée au sein de l'équipe afin de comparer les résultats du planificateur *GTOHP* avec ceux d'une version Java du planificateur HTN *SHOP*. Le choix s'est porté sur cette version Java de l'algorithme *SHOP* afin de reproduire exactement les mêmes conditions pour les deux algorithmes, en ayant notamment les deux planificateurs codés dans le même langage. Les tests ont été menés sur des versions HTN de 12 domaines de planification [5] utilisés dans les *deterministic tracks* des compétitions internationales de planification. Ces domaines comportent une palette variées de types de problèmes, notamment des problèmes de composition, d'exploration de chemins, d'énumération, avec et sans tâches répétitives. Diversifier les problèmes permet d'évaluer les performances des deux algorithmes pour chaque type de problèmes.

L'implémentation de la version HTN de ces domaines a été répartie sur deux catégories d'utilisateurs : Les domaines *Barman*, *Child-Snack*, *Rover* et *Satellite* ont été implémentés par un utilisateur avec des connaissances très avancées dans la programmation informatique et la planification HTN. Le reste des domaines a été implémenté par un utilisateur avec des connaissances avancées en programmation informatique et débutant dans la planification HTN.

En sachant que *GTOHP* et *SHOP* utilisent deux langages de définition des domaines et des problèmes différents [6], l'implémentation des versions HTN de ces domaines respecte les règles suivantes :

— La partie classique du domaine, *i.e.* les types, les prédicats et les opérateurs de la version HTN correspondent exactement à ceux de la version originale, utilisée dans les compétitions.
— Les problèmes de la version HTN correspondent exactement à ceux de la compétition, en y ajoutant les tâches à planifier.
— Les méthodes de décompositions définies dans les deux domaines sont définies de la même façon afin de ne pas introduire de biais dans l'évaluation. Cependant, quelques différences mineurs, liées à la différence entre les deux langages peuvent subsister sans que cela n'affecte les décompositions.

5. Les domaine utilisés sont : *Barman, Blocksworld, Depots, Elevator, Gripper, Hiking, Movie, Parking, Zenotravel, Childsnack, Rover, et Satellite*

6. *GTOHP* utilise HPDDL qui est une version HTN du langage PDDL, alors que *SHOP* utilise un langage spécifique présenté dans (Nau et al. , 2006)

Les résultats du planificateur *GTOHP* ont aussi été comparés à ceux du planificateur *Fast Downward* (Helmert, 2006) sur quatre domaines de planification présentant des spécificités différentes, cela afin d'obtenir des points de comparaison avec un planificateur classique récent. Ces quatre domaines sont : *Barman, Child-Snack*, *Rover* et *Satellite*.

4.4.1.A Matériel

Tous les tests ont été effectués sur un ordinateur doté d'un processeur Intel Core i7 cadencé à 2.2 Ghz et de 16 Go de RAM DDR3 avec 1600MHz. Le nombre de cœurs du processeur n'affecte pas les résultats car les deux algorithmes n'implémentent pas de techniques de parallélisation et le processus java est exécuté sur un seul cœur. Les autres cœurs du CPU sont utilisés par la JVM uniquement pour la compression de données si la mémoire de 10 Go allouée au processus est dépassée.

4.4.1.B Critères d'évaluation

L'évaluation de l'algorithme d'instanciation et de simplification se concentre sur trois critères :
La simplification : Le premier critère d'évaluation permet de voir à quel point les problèmes HTN sont simplifiés en termes de nombre d'instances de méthodes. Pour cela, on compare le nombre de décompositions réellement obtenues après le processus d'instanciation et de simplification avec le nombre d'instances de méthode possibles définies dans les domaines de planification. Afin de quantifier cela, on prend en compte le taux de simplification qui se calcule, pour chaque problème, en divisant le nombre de méthodes simplifiées [7] par le nombre d'instances possibles. Prenons comme exemple, un problème produisant 50000 instances de méthodes sans simplification, avec un taux de simplification de 70%, le nombre de méthodes simplifiées est alors de $50000 * 0,7 = 35000$ instances de méthode, laissant alors 15000 décompositions pertinentes pour le problème en question.

Afin de mesurer la simplification réalisée, le taux de simplification doit être transposé au nombre global des instances de méthodes non simplifiées. Par exemple, si le taux de simplification est stable et que le nombre d'instances possibles augmente avec la complexité des problèmes, alors le nombre des méthodes simplifiées augmente lui aussi de façon proportionnelle.

Le temps de calcul : L'objectif premier du processus d'instantiation étant l'amélioration des per-

7. Les méthodes simplifiées sont les méthodes qui ont été rejeté et effacé du problème car elles ne sont pas pertinentes et ne conduisent jamais à un plan solution

formances des planificateurs HTN, nous cherchons donc à vérifier que la simplification des problèmes de planification améliore réellement les performances des planificateurs HTN. Pour cela, le deuxième critère d'évaluation pris en compte est le temps de traitement utilisé par chaque planificateur pour trouver le premier plan solution. Dans le cas du planificateur *GTOHP*, le temps de traitement est calculé comme suit :

$$t_{total} = t_{grounding} + t_{research}$$

Où t_{total} est le temps total de traitement, $t_{grounding}$ est le temps utilisé par le processus d'instanciation et de simplification, et $t_{research}$ est le temps utilisé par le processus de recherche. Cette décomposition du temps de traitement permet de mettre en avant le temps de traitement de chaque processus et permet une meilleure analyse et une évaluation plus détaillée du processus global.

En plus du temps de traitement, nous considérons aussi pour chaque planificateur un score obtenu pour chaque domaine en nous fondant sur la formule utilisée dans la catégorie *agile* des compétitions de planification. Dans cette catégorie, chaque planificateur obtient un *score* pour chaque problème, calculé à partir de son propre temps comparé au meilleur temps des autres planificateurs comme suit :

$$score = 1 \quad Si\ le\ temps\ de\ traitement < 1\ sec$$

$$score = 1\ /\ (1 + \log(t/t_{min})) \quad Sinon$$

Cette formule permet de masquer les différences lorsque le temps de traitement est inférieur à 1 sec et de calculer le score des planificateurs en fonction du plus performant d'entre eux. Le score d'un planificateur est égal à 1 s'il est le meilleur. Plus son temps est supérieur au meilleur temps plus son score approche de 0, jusqu'à l'atteindre s'il ne trouve pas de solution.

La longueur des plans : La longueur d'un plan correspond au nombre d'actions qu'il contient. Même si, en HTN et sans heuristique, ce critère dépend surtout de la définition des domaines, il reste néanmoins incontournable dans une évaluation d'un planificateur quel qu'il soit. Sans pondération ou coût sur les actions, on cherche alors à minimiser le nombre d'actions à réaliser. Nous avons alors comparé la longueur des plans obtenus avec les trois algorithmes : *GTOHP, SHOP* et *Fast Downward*.

On peut noter que la taille de la mémoire utilisée n'a pas été retenue en tant que critère d'évaluation. Cela est dû au fait qu'elle dépend beaucoup de la qualité du code, même si les deux planificateurs sont codés en Java. En plus, lors de l'expérimentation, elle a été limitée à 10 Go et si un planificateur dépasse cette limite, il se retrouve grandement pénalisé en terme de temps d'exécution. En effet, lorsqu'un programme Java dépasse la taille de la mémoire qui lui a été attribuée, la JVM crée

de la mémoire virtuelle en lançant un processus de *swapping* chronophage qui consomme énormément de ressources, il en résulte en géneral une erreur de dépassement de mémoire si ce dernier est trop grand.

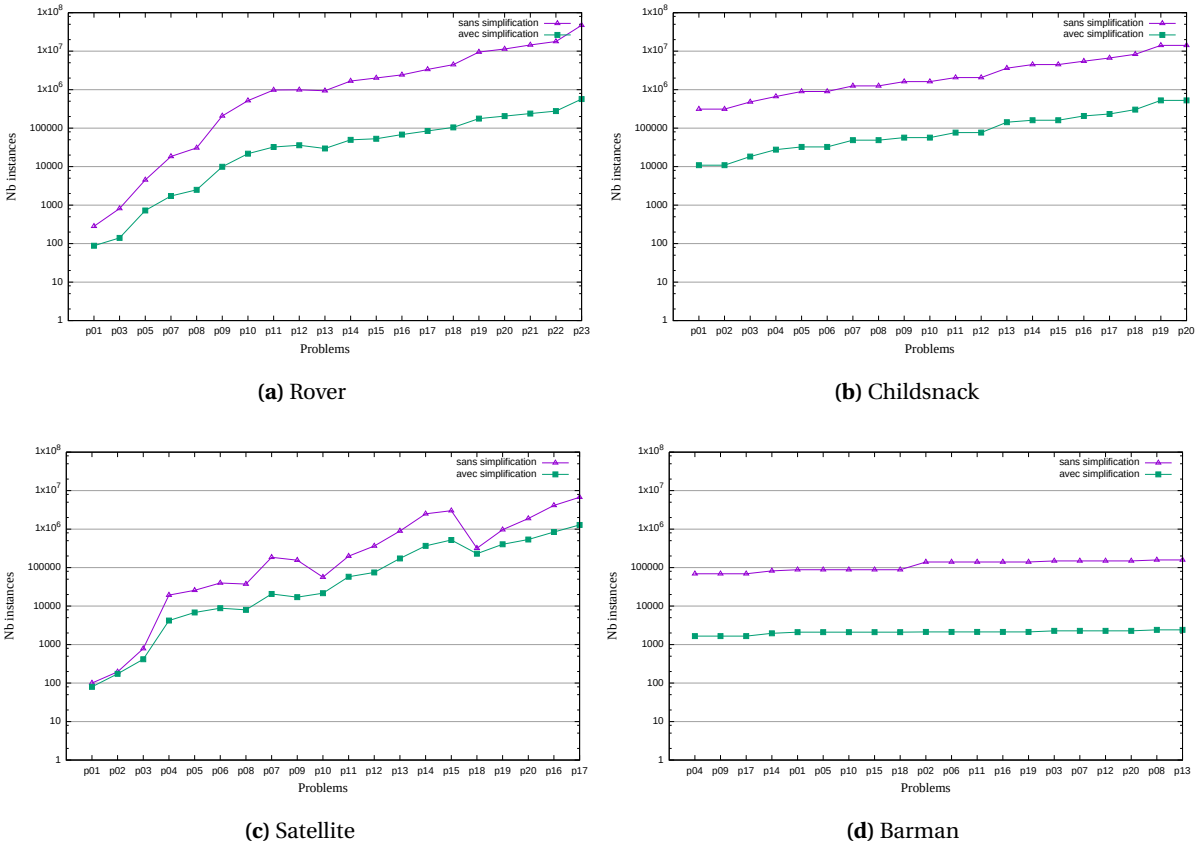
4.4.2 Évaluation du taux de simplification

Les résultats de l'évaluation de l'algorithme d'instanciation et de simplification en fonction du critère de simplification sont présentés sous forme de graphiques désignant le nombre total d'instances des méthodes pour chaque problème ainsi que le nombre de décompositions produites par le processus d'instanciation et de simplification. L'axe des ordonnée représente le nombre d'instances, alors que l'axe des abscisses représente les différents problèmes du domaine triés par ordre croissant en fonction du nombre d'objets, sans distinction entre leurs types. Le taux de simplification d'un problème est facilement calculable en divisant la différence entre les deux valeurs obtenues pour ce problème par le nombre d'instances de méthodes possibles pour le même problème.

Il faut noter que le nombre total d'instances n'est jamais généré lors du processus d'instanciation et de simplification et que l'élagage est effectué en amont de la génération de la décomposition. La représentation est faite sur une échelle logarithmique en base 10 afin de préserver les détails des valeurs des deux graphiques dans les petites et les grandes valeurs. Il faut bien retenir en interprétant les graphiques que le nombre d'instances dans un palier de l'échelle logarithmique décimale est multiplié à chaque fois par dix dans le suivant.

La figure 4.5 présente le nombre d'instances et de décompositions résultants de l'instanciation et simplification des domaines : *Rover, Childsnack, Satellite, et Barman*. On remarque dans les graphiques que la différence entre le nombre de décompositions générées après simplification et le nombre d'instances possibles est très important spécialement dans les problèmes de grande taille. Dans la figure 4.5 (a, b et c), le nombre d'instances des méthodes augmente de façon très rapide avec l'augmentation de la taille des problèmes. Dans la figure 4.5 (a) par exemple, le taux de simplification est de 84% dans le problème 5 avec 4 520 instances de méthodes et 724 décompositions pertinentes, alors qu'il atteint les 99% dans le dernier problème, puisque l'instanciation et la simplification produit 568 850 décompositions sur 46 980 192 instances de méthodes possibles.

La même augmentation du nombre d'instances simplifiées est observée dans la figure 4.5 (c), on note cependant, une baisse de ce nombre dans les problèmes 10 et 18 malgré une augmentation du nombre d'objet. Cela est dû au types d'objets déclarés dans le problème. Par exemple, Dans le problème 10 par rapport au problème 9 de satellite, on a une diminution du nombre d'objets de type instrument contre une augmentation des objets de types direction. Cette diminution est expliquée



(a) Rover **(b)** Childsnack

(c) Satellite **(d)** Barman

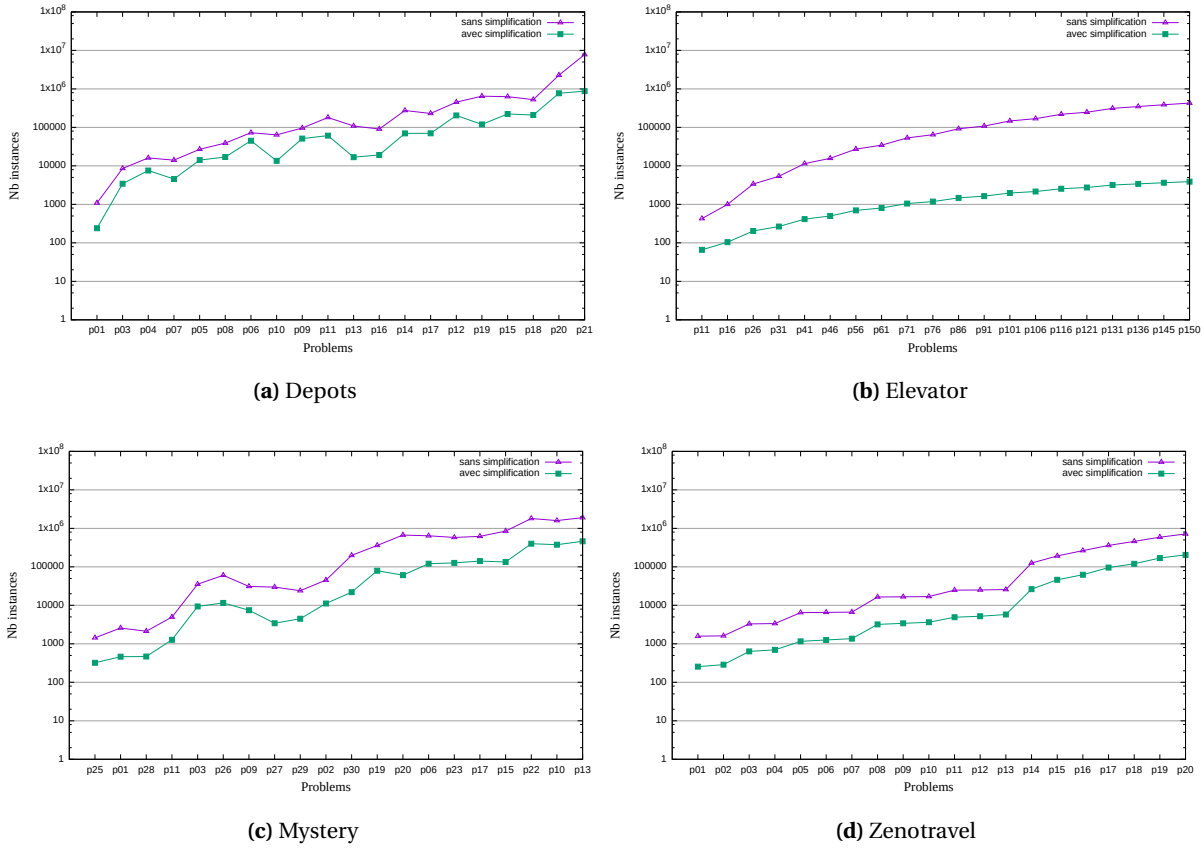
FIGURE 4.5 – Graphiques représentant le nombre d'instances et de décompositions des domaines : (a) Rover, (b) Childsnack, (c) Satellite et (d) Barman sur une échelle logarithmique

par l'apparition du type instrument dans plus de méthodes et d'inerties que le type direction. Dans la figure 4.5 (d), le taux de simplification est élevé mais reste stable au tour des 98%. Le nombre de méthodes simplifiées aussi est très élevé mais stable, et cela est dû à la taille stable des problèmes de *Barman* qui contiennent entre 37 et 43 objets.

La figure 4.6 présente le nombre d'instances et de décompositions résultants de l'instanciation et la simplification des domaines : *Depots, Elevator, Mystery, et Zenotravel*. On peut observer deux types de résultats. Dans les figures 4.6(b et d), les observations restent les mêmes que dans la figure précédente avec un taux de simplification élevé, entre 90-99% [8] dans (b), et entre 71-84% dans (d) en présentant une tendance baissière dans le domaine *Zenotravel* (d), même si le nombre de méthodes simplifiées augmente en passant de 1 326 dans le premier problème de *Zenotravel* à 510 510 dans le problème 20. Dans les figures 4.6(a et c), le taux de simplification varie beaucoup par rapport à celui de (b et d), avec une moyenne de 66% dans (a) et 80% dans (b). Le nombre de méthodes simplifiées

8. Si on ne considère pas le problème 0 qui présente un taux de simplification de 36% 8 instances simplifiées sur 22

augmente d'un problème à un autre, en passant de 846 dans le premier problème à 6 926 920 dans le dernier problème du domaine *Depots*, et de 1 110 dans le premier problème à 1 437 961 dans le dernier problème de *Mystery*. L'instabilité du taux de simplification dans ces deux domaines est due, comme dans le domaine satellite, à la variation du nombre d'objets par type, malgré une augmentation continue du nombre d'objets d'un problème à un autre.



(a) Depots

(b) Elevator

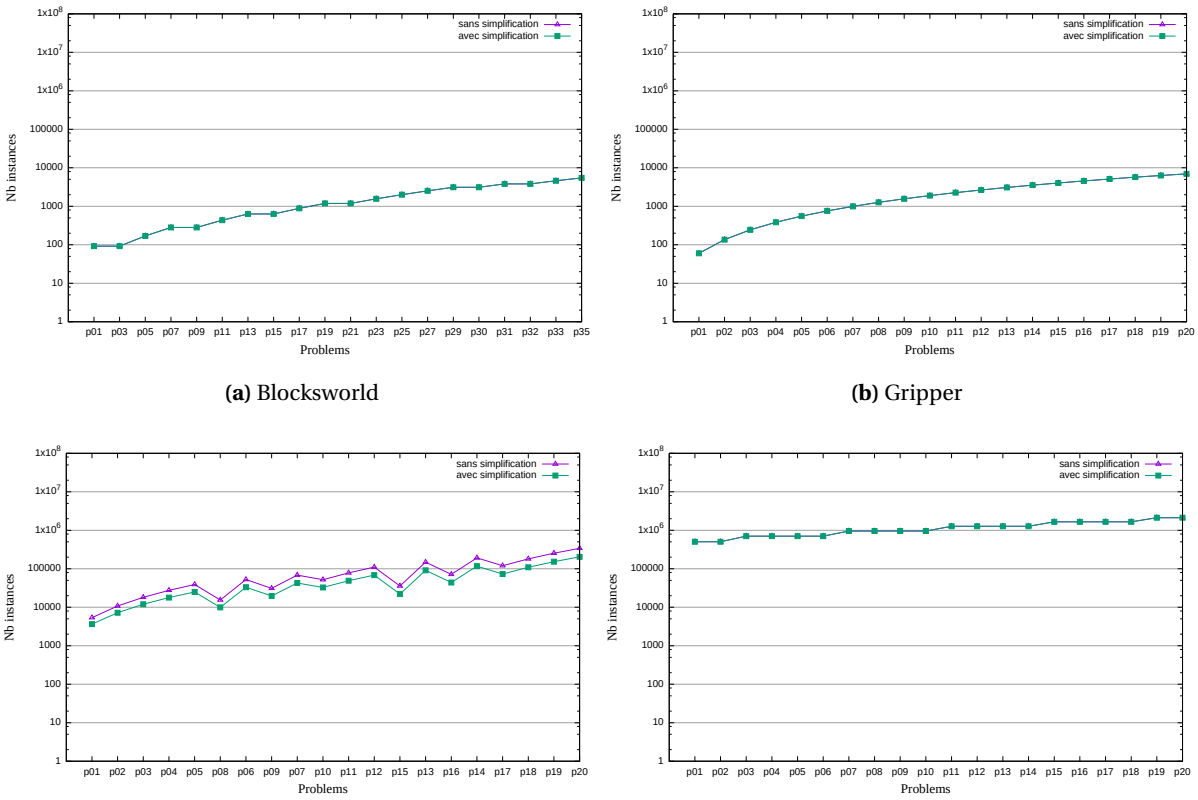
(c) Mystery

(d) Zenotravel

FIGURE 4.6 – Graphiques représentant le nombre d'instances et de décompositions des domaines : (a) Depots, (b) Elevator, (c) Mystery et (d) Zenotravel sur une échelle logarithmique

La figure 4.7 présente le nombre d'instances et de décompositions résultants de l'instanciation et la simplification des domaines : *Blocksworld, Gripper, Hiking, et Parking*. Dans la figure 4.7(c), on peut voir que le taux de simplification est assez petit par rapport aux figures précédentes. Il varie entre 32 et 42% avec un nombre d'instances simplifiées variant entre 1 710 et 136 262. Cependant, l'observation la plus importante qui peut être faite sur les résultats présentés est que le nombre d'instances des méthodes et celui des décompositions pertinentes sont exactement les mêmes dans les figures 4.7(a, b, et d). Lorsque le nombre de décompositions est égal au nombre d'instances de méthodes, cela veut dire que le nombre de méthodes simplifiées est égal à 0 et que le taux de simplification est

de 0%. L'explication de ce résultat, qui peut paraître négatif au premier abord, est liée directement à la définition du domaine de planification. Comme cela a été expliqué dans le chapitre 4, l'étape clé dans l'algorithme de simplification est le calcul des inerties, et toutes les simplifications qui suivent dépendent directement ou indirectement du résultat de cette étape et de l'utilisation de ces inerties dans les pré-conditions et les effets des opérateurs et des méthodes. Les domaines *Blocksworld, Gripper, et Parking* ne définissent aucun prédicat qui peut être considéré comme une inertie. Donc quelque soit l'état initial définit dans les problèmes, toutes les instances de méthodes et d'opérateurs peuvent mener à une solution mais ne peuvent pas être simplifiées.



(a) Blocksworld

(b) Gripper

(c) Hiking

(d) Parking

FIGURE 4.7 – Graphiques représentant le nombre d'instances et de décompositions des domaines : (a) Blocksworld, (b) Gripper, (c) Hiking, (d) Parking sur une échelle logarithmique

On peut observer dans les chiffres d'instanciation et simplification des domaines *Blocksworld et Gripper* que leurs complexité est assez basse en comparaison aux domaines précédents. Le domaine *Parking* présente une complexité du même ordre de grandeur que les derniers problèmes simplifiés dans les domaines *Rover, Childsnack, Satellite, Depots, Mystery, ou Zenotravel*. Donc, malgré l'absence de simplification pour ces domaines, on observe, lors de l'évaluation des performances de recherche,

que l'algorithme HTN instancié réalise de très bonnes performances, grâce notamment à la faible complexité de ces domaines.

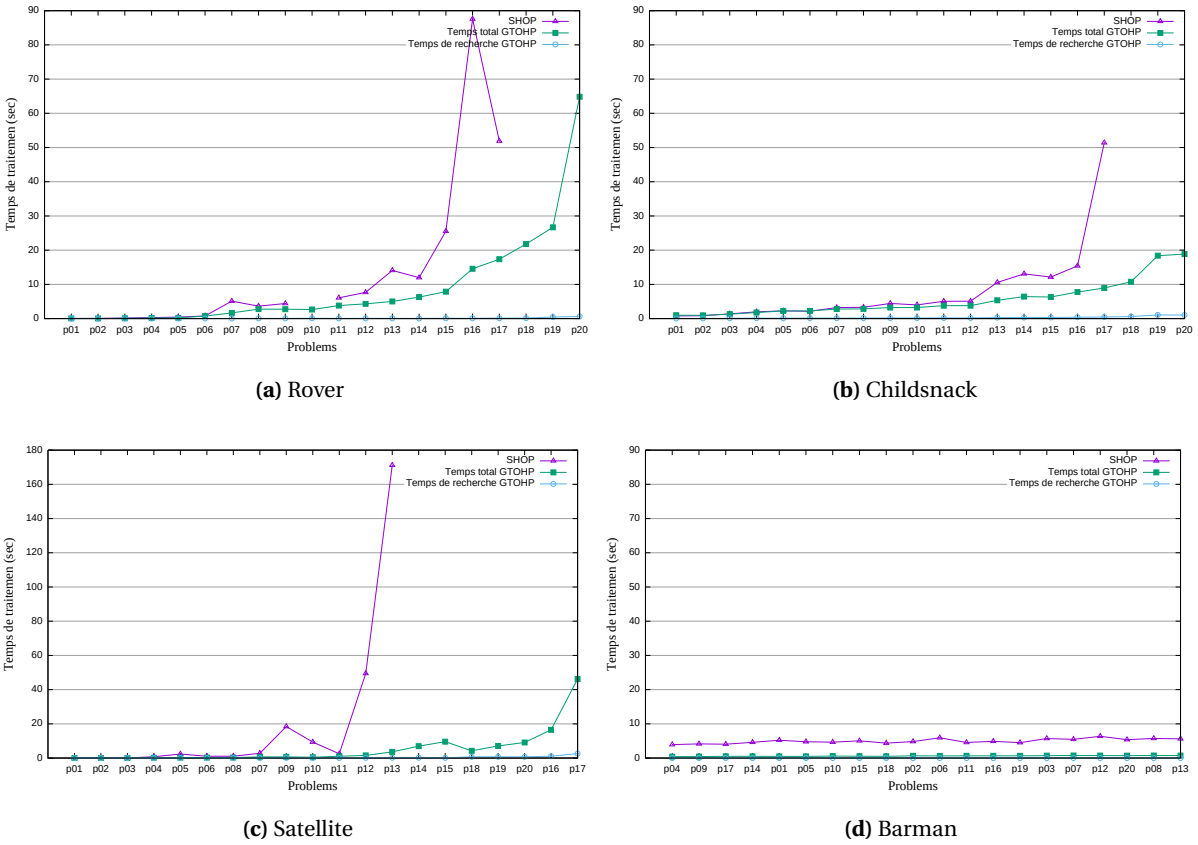
De manière générale, les domaines qui définissent le plus d'inerties sont ceux qui définissent l'environnement du problème et plus précisément les liens entre les objets du domaines. Ce sont ces liens qui apportent une complexité supplémentaire aux problèmes, comme par exemple, les liens d'accessibilité entre les *waypoints* du domaine *Rover* qui rajoutent de la complexité au processus de planification de trajectoire. Autrement dit, moins il y a de liens entre les objets, moins on a besoin de citer d'objets dans les méthodes et les opérateur et donc, moins les problèmes sont complexes.

On montre à partir des résultats présentés précédemment que le processus d'instanciation permet de simplifier de façon importante les problèmes de planification HTN en réduisant de façon drastique le nombre de décomposition qu'ils contiennent. On montre que ces performances sont directement liées à la définition du domaine de planification et à la complexité des problèmes, plus les domaines et les problèmes sont complexes, plus le processus permet de simplifier des décompositions qui ne sont pas pertinentes. En sachant que la diminution du nombre des décompositions dans un problème induit une diminution du facteur de branchement dans l'arbre de recherche, on montre, dans le prochain paragraphe que cela améliore les performances des algorithmes de planification HTN.

4.4.3 Évaluation du temps de calcul

Tous les résultats de l'évaluation du planificateur *GTOHP* en fonction du temps de recherche sont présentés dans les figures 4.8, 4.9 et 4.10. L'axe des abscisses représente les problèmes de planification triés par ordre croissant en fonction du nombre d'objets. L'axe des ordonnées représente le temps de traitement (en secondes) nécessaire pour trouver un plan solution. Chaque graphique comprend trois tracés, un pour le temps de traitement total pour *GTOHP*, un autre pour le temps de recherche qui est égal au temps total moins le temps du processus d'instanciation et de simplification, le dernier représente le temps de traitement total du planificateur *GTOHP*. Les résultats du temps de traitement sont affichés dans la même échelle décimale de 0 à 90 sauf pour les domaines *Satellite et Parking* qui comportent des valeurs supérieures à 90 secondes. Le temps de recherche maximum pris en compte est de 10 min (soit 600 sec) et l'absence d'une valeur pour un problème signifie que le planificateur n'a pas été capable de trouver une solution dans le temps alloué.

Dans la figure 4.8 qui présente les résultats des domaines *Rover, Childsnack, Satellite et Barman*, *GTOHP* est le plus rapide pour trouver une solution dans tous les problèmes. Cela permet de confirmer que l'instanciation et la simplification améliore de façon importante les performances des pla-



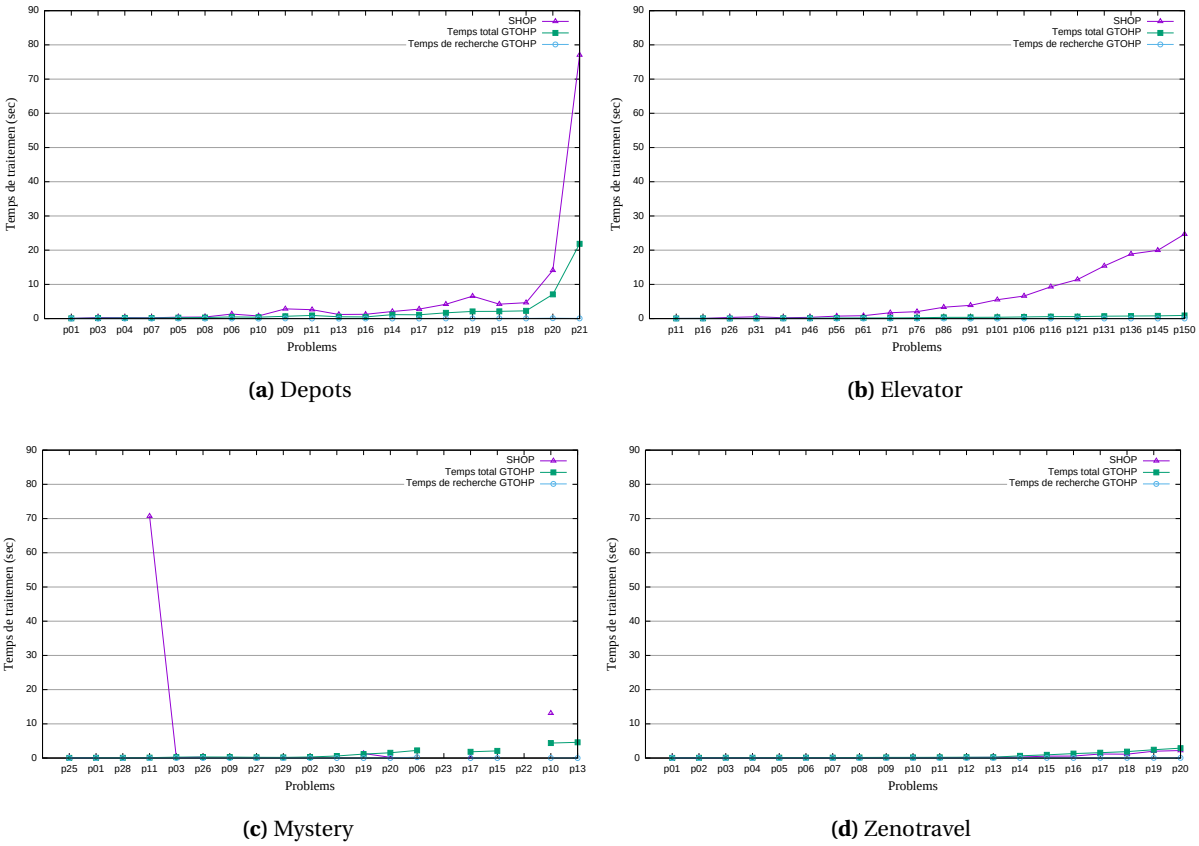
(a) Rover **(b)** Childsnack

(c) Satellite **(d)** Barman

FIGURE 4.8 – Graphiques représentant le temps de traitement nécessaire afin trouver un plan solution pour les problèmes des domaines : (a) Rover, (b) Childsnack, (c) Satellite et (d) Barman

nificateurs HTN. De plus, dans tous les problèmes testés, le temps de recherche représente moins de 5% du temps de traitement total de *GTOHP*, sans dépasser les 3 secondes, même pour des problèmes relativement complexes. Ceci peut s'avérer être très intéressant dans les cas de re-planification, lors des changements de buts par exemple. Du fait que le processus d'instanciation et de simplification n'est affecté que par des changements dans l'état initial, les opérateurs ou les méthodes, aucun processus d'instanciation et de simplification n'est nécessaire pour re-planifier des problèmes impliquant uniquement des changements de buts. Cependant, en cas de changements dans l'état initial, les opérateurs ou les méthodes, un processus d'instanciation et de simplification est nécessaire afin de garantir l'intégrité du problème instancié.

Dans les graphiques a, b et c de la figure 4.8, la différence de temps de recherche entre les deux algorithmes est très faible pour les petits problèmes, et croît rapidement avec l'accroissement de la taille de ces derniers. Dans le domaine Barman, la différence de temps de recherche est stable (autour de 4 secondes), cela est dû, comme pour le taux de simplification, à la stabilité du nombre d'objets



(a) Depots

(b) Elevator

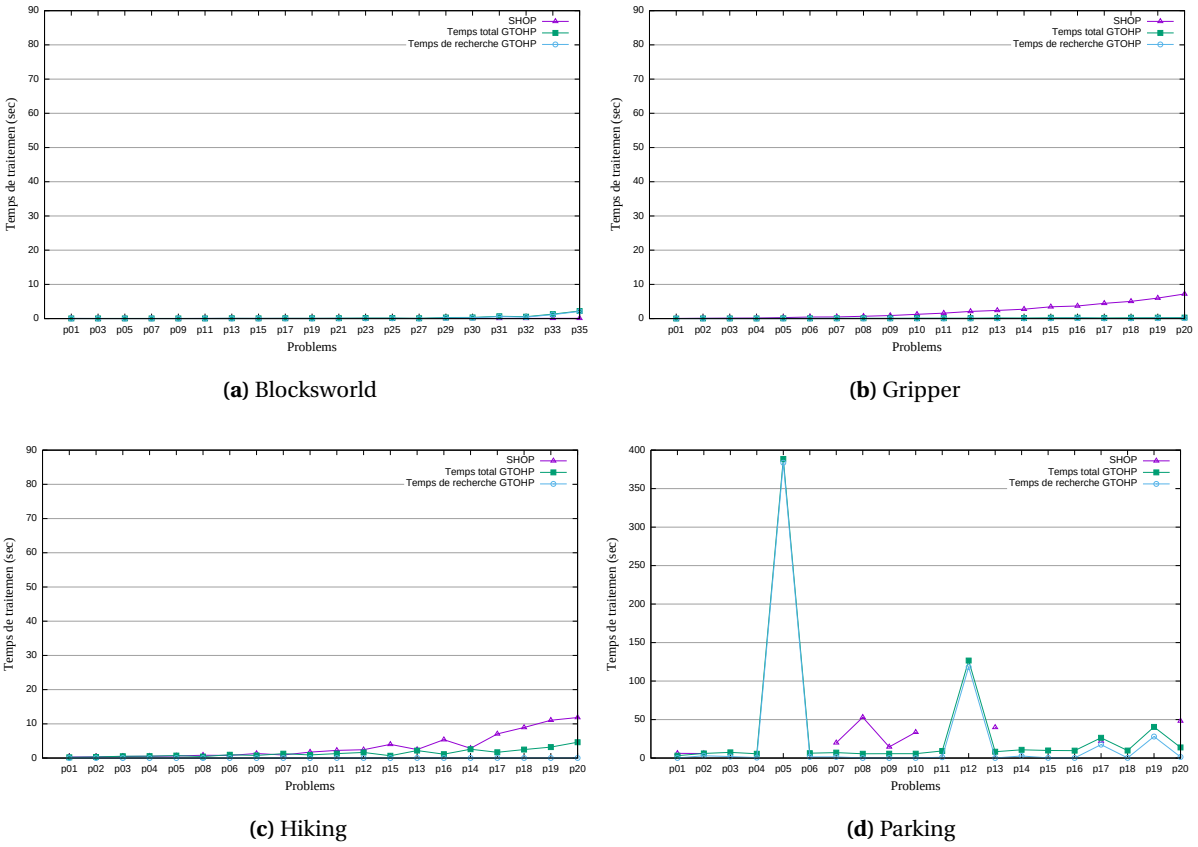
(c) Mystery

(d) Zenotravel

FIGURE 4.9 – Graphiques représentant le temps de traitement nécessaire afin trouver un plan solution pour les problèmes des domaines : (a) Depots, (b) Elevator, (c) Mystery et (d) Zenotravel

dans les problèmes de *Barman*. On note aussi que *SHOP* ne trouve pas de solution, dans le temps alloué, à partir du problème 19, dans le domaine *Rover*, du problème 16 dans *Childsnack* et du problème 12 dans *Satellite*, alors que *GTOHP* trouve des solutions pour tous les problèmes dans le temps alloué. Il faut 76.33 sec pour trouver un plan pour le plus gros problème de *Rover* avec 7844 décompositions, 21.19 sec pour le plus grand problème de *Childsnack* avec 25 décompositions, et 51.77 sec pour le plus gros problème de satellite avec 64550 décompositions.

Les résultats affichés dans les figures 4.9 et 4.10 confirment les bons résultats de *GTOHP* par rapport à ceux de *SHOP* avec des performances nettement supérieures en terme de temps de traitement. Cela se voit nettement dans les résultats des domaines *Depots et Elevator* où l'écart entre les deux planificateurs augmente de façon considérable avec l'augmentation de la complexité des problèmes, atteignant 35,26 sec dans le problème le plus complexe de *Depots* et 23,77 sec dans celui de *Elevator*. Cette différence est moins importante dans les domaines *Gripper et Hiking* mais elle montre une tendance haussière atteignant 6,85 sec dans *Gripper* et 7,05 sec dans *Hiking*. La différence de per-



(a) Blocksworld

(b) Gripper

(c) Hiking

(d) Parking

FIGURE 4.10 – Graphiques représentant le temps de traitement nécessaire afin trouver un plan solution pour les problèmes des domaines : (a) Blocksworld, (b) Gripper, (c) Hiking et (d) Parking

formance entre les deux planificateurs est beaucoup plus prononcée dans les domaines *Mystery et Hiking* représentés respectivement dans les figure 4.9(c) et 4.10(d). Dans *Mystery*, *SHOP* ne trouve que 11 solutions sur 20 alors que *GTOHP* a trouvé des solutions pour tous les problèmes exceptés les problèmes 22 et 23 qui peuvent, cependant, représenter des problèmes insolubles. Dans *Parking*, le bilan est encore plus positif pour *GTOHP* puisque *SHOP* ne trouve que 9 solutions sur 20 contre 20/20 pour *GTOHP*.

Dans *Zenotravel*, dont les problèmes peuvent être considérés comme étant simples [9], *SHOP* est globalement plus performent par rapport à *GTOHP* en terme de temps de traitement total, avec une différence maximum de 2,46 sec. Cependant, *GTOHP* affiche un temps de recherche n'excédant pas les 0,06 sec et tous le reste représente le temps de traitement du processus d'instanciation et de simplification. Considérant les temps de traitement très courts pour ce domaine, la comparaison entre

9. Les problèmes sont considérés comme faciles, avec un temps de recherche n'excédant pas les 2,82 sec pour les deux planificateurs. Cela est dû à la définition HTN domaine qui guide les algorithmes de recherche de manière très efficace

les deux planificateurs sera plus pertinente dans le tableau des scores 4.3.

Le tableau 4.2 récapitule, selon la formule définie dans 4.4.1.B, les scores obtenus par les planificateurs *Fast-Downward, GTOHP et SHOP* dans les domaines HTN définis par un utilisateur expert, à savoir *Rover, Childsnack, Satellite et Barman*. Dans le domaine *Rover*, *Fast-Downward* obtient un score parfait de 20/20. Cela signifie que *Fast-Downward*a été le plus rapide pour trouver un plan pour tous les problèmes. *GTOHP* est deuxième avec un score de 16,87/20 et *SHOP* est troisième avec un score de 12,07. Donc *GTOHP* est en moyenne 15,63% moins efficace que *Fast-Downward*, et 24% plus efficace que *SHOP*. En revanche il est le plus efficace dans le domaine *Childsnack* avec un score de 18.89/20 et est respectivement 63,85% et 19,88% plus efficace que *Fast-Downward* et *SHOP*.

Les bons scores obtenus par les planificateurs HTN en comparaison à *Fast-Downward*, sur le domaine *Childsnack*, montre l'efficacité de la planification HTN dans les problèmes nécessitant de nombreux *backtracks*, les méthodes de décomposition permettant de restreindre fortement l'espace de recherche. Dans le domaine *Satellite*, *GTOHP* obtient un score de 19.22/20 et est 10.98% plus efficace que *Fast-Downward* avec un score de 16.8, et 52.14% plus efficace que *SHOP* avec un score de 7.75. Dans *Barman* aussi, *GTOHP* obtient 20/20 avec tous les plans trouvés en moins d'une seconde. *SHOP* est moitié moins efficace avec un score de 10,51. *Fast-Downward* obtient un très mauvais score de 0.65, puisqu'il ne trouve que deux plans sur 20. Le score global obtenu dans ces quatre domaines est de 42,3/80 pour *Fast-Downward*, de 74,99/80 pour *GTOHP* et 44,85/80 pour *SHOP*. Donc, *GTOHP* est 40,86% plus efficace que *Fast-Downward* et 37,67 % plus efficace que *SHOP*.

Le tableau 4.3 rassemble les scores obtenus par les planificateurs HTN *GTOHP et SHOP* dans les domaines HTN définis par un utilisateur débutant, à savoir *Blocksworld, Depots, Elevator, Gripper, Hiking, Mystery, Parking et Zenotravel*. *GTOHP* obtient un score parfait de 20/20 dans les domaines *Depots, Elevator et Gripper* contre respectivement 16,35 , 13,51 et 14,01 pour *SHOP*. Il obtient aussi un meilleur résultat que *SHOP* dans les domaines *Hiking, Mystery et Parking* avec respectivement 19,87, 17,06 et 19,91 pour *GTOHP* et 17,03 , 9,44 et 6,48 pour *SHOP*. Ce dernier obtient tout de même un meilleure score que *GTOHP* dans les domaines *Blocksworld et Zenotravel* avec un score parfait de 20/20 contre 19,06 et 18,82 pour *GTOHP*. Pour ces deux derniers domaines, les scores donnent plus de détails que les graphiques et permettent de voir que *GTOHP* est moins performant que *SHOP* dans 8 problèmes sur 40. Il est, aussi dans ces deux domaines, en moyenne, 5,27% moins performant que *SHOP*. En sachant que ces deux problèmes sont très simples et que le temps de traitement maximum est inférieur à 3 sec, on ne peut pas considérer les résultats de *GTOHP* comme mauvais.

	Rover			Childsnack			Satellite			Barman		
Pb	FD	GTOHP	SHOP	FD	GTOHP	SHOP	FD	GTOHP	SHOP	FD	GTOHP	SHOP
01	1	1	1	0	1	1	1	1	1	0	1	0.51
02	1	1	1	0.49	1	1	1	1	1	0	1	0.55
03	1	1	1	0.34	1	0.99	1	1	1	0.38	1	0.52
04	1	1	1	1	0.62	0.61	1	1	1	0	1	0.51
05	1	1	1	1	0.66	0.66	1	1	0.33	0	1	0.51
06	1	1	1	1	0.71	0.73	1	1	0.39	0	1	0.53
07	1	0,75	0,55	0	1	0.96	1	1	0.37	0	1	0.54
08	1	0,76	0,70	0	1	0.99	1	1	0.38	0	1	0.53
09	1	0,87	0,74	0	1	0.91	1	1	0.33	0	1	0.51
10	1	0,99	03	1	0.89	0.84	1	1	0.42	0	1	0.51
11	1	0,72	0,63	0	1	0.94	0.84	1	0.75	0	1	0.54
12	1	0,77	0,64	0	1	0.95	1	0.96	0.40	0	1	0.51
13	1	0,81	0,59	0	1	0.83	0.71	1	0.38	0	1	0.53
14	1	0,76	0,62	0	1	0.81	0.53	1	0	0.27	1	0.51
15	1	0,75	0,54	0	1	0.84	0.64	1	0	0	1	0.50
16	1	0,74	0,47	0	1	0.83	0.42	1	0	0	1	0.54
17	1	0,74	0,54	0	1	0.59	0	1	0	0	1	0.52
18	1	0,71	07	0	1	0	1	0.66	0	0	1	0.52
19	1	0,70	04	0	1	0	0.66	1	0	0	1	0.56
20	1	0,72	0	0	1	0	1	0.60	0	0	1	0.54
	20	16,87	12,07	4.84	18.89	14.51	16.81	19.22	7.75	0.65	20	10.51

TABLE 4.2 – Scores des planificateurs *GTOHP*, *SHOP* et *Fast-Downward* selon l'agile track de l'IPC dans les domaines *Rover*, *Childsnack*,*Satellite* et *Barman*

D'un point de vue global sur les domaines *Blocksworld, Depots, Elevator, Gripper, Hiking, Mystery, Parking et Zenotravel*, *GTOHP* obtient un score global de 154,75 contre 117,27 pour *SHOP*. En considérant le score maximal de 160 comme étant 100%, *GTOHP* est 23,42% plus performant que *SHOP*. Il ne serait pas pertinent de comparer les scores du premier tableau avec le second, parce que ceux du premier sont fortement influencés par les résultats de *Fast-Downward*, cependant en enlevant les résultats de ce dernier, *GTOHP* est 28,8% plus performant que *SHOP* sur l'ensemble des domaines.

Tous les résultats des expérimentations réalisées, qui ont été présentés sous forme graphique et sous forme numérique, confirment bien que la planification HTN avec instanciation et simplification est plus efficace en terme de temps d'exécution que la planification HTN classique avec un net avantage sur les problèmes complexes.

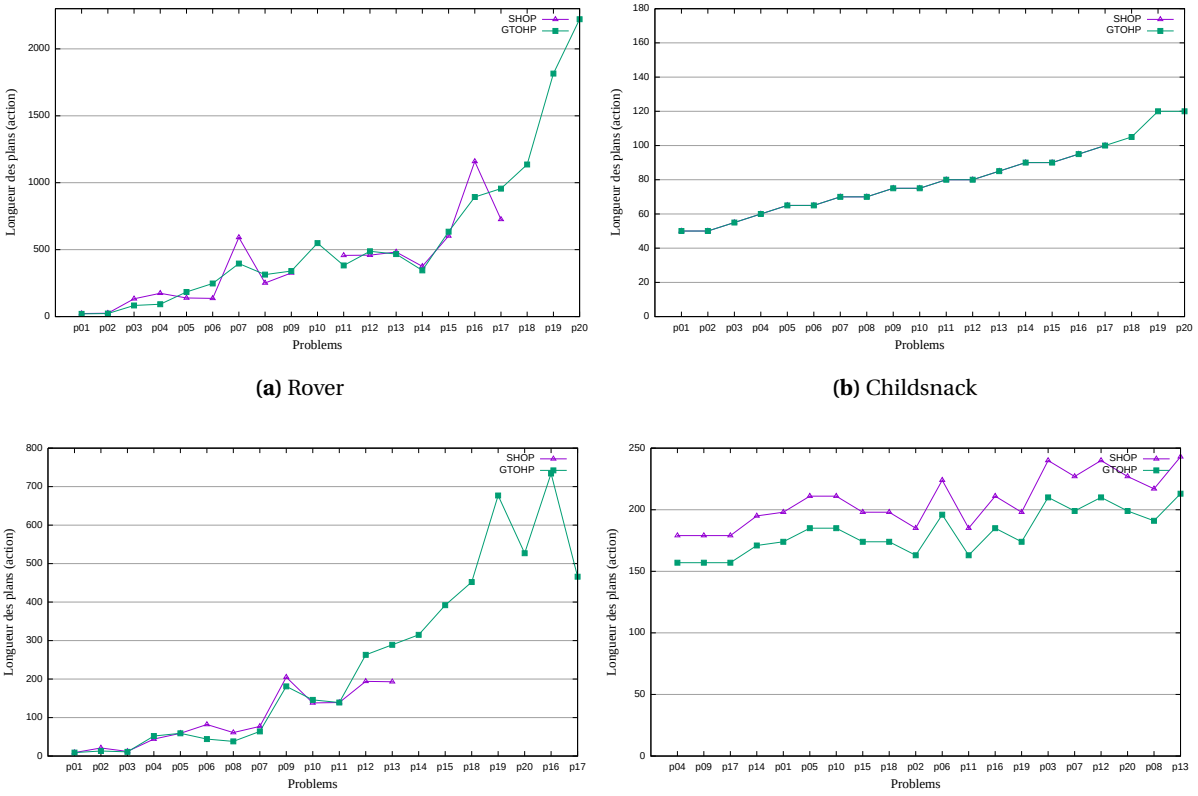
	Blocksworld		Depots		Elevator		Gripper		Hiking		Mystery		Parking		Zenotravel	
Pb	GTOHP	SHOP	GTOHP	SHOP	GTOHP	SHOP	GTOHP	SHOP	GTOHP	SHOP	GTOHP	SHOP	GTOHP	SHOP	GTOHP	SHOP
01	1	1	1	1	1	1	1	1	1	1	1	1	1	0,77	1	1
02	1	1	1	1	1	1	1	1	1	1	1	1	0,98	1	1	1
03	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
04	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1
05	1	1	1	0,69	1	1	1	1	1	1	1	0	1	0	1	1
06	1	1	1	1	1	1	1	1	1	1	1	0,66	1	0	1	1
07	1	1	1	1	1	1	1	1	0,87	1	1	0,25	1	0,68	1	1
08	1	1	1	0,61	1	1	1	1	1	1	1	0	1	0,50	1	1
09	1	1	1	1	1	0,52	1	1	1	0,78	1	0	1	0,70	1	1
10	1	1	1	0,69	1	0,51	1	0,49	1	0,80	0,53	1	1	0,56	1	1
11	1	1	1	0,71	1	0,49	1	0,47	1	0,81	1	0,95	1	0	1	1
12	1	1	1	0,73	1	0,48	1	0,45	1	0,87	0,53	1	1	0	1	1
13	1	1	1	0,77	1	0,46	1	0,45	1	0,93	0	0	1	0,59	1	1
14	1	1	1	0,77	1	0,45	1	0,44	1	0,98	0	0	1	0	1	1
15	1	1	1	0,70	1	0,44	1	0,44	1	0,57	1	1	1	0	0,79	1
16	1	1	1	0,69	1	0,43	1	0,44	1	0,61	1	1	1	0	0,74	1
17	1	1	1	0,73	1	0,42	1	0,46	1	0,63	1	1	0,92	1	0,86	1
18	1	1	1	0,65	1	0,41	1	0,45	1	0,64	1	0	1	0	0,81	1
19	0,43	1	1	0,75	1	0,42	1	0,42	1	0,65	1	0	1	0	0,93	1
20	0,39	1	1	0,79	1	0,41	1	0,43	1	0,71	1	0	1	0,64	0,89	1
	18,82	20	20	16,35	20	13,51	20	14,01	19,87	17,03	17,06	9,44	19,91	6,48	19,06	20

TABLE 4.3 – Scores des planificateurs *GTOHP* et *SHOP* selon l'agile track de l'IPC dans les domaines *Blocksworld, Depots, Elevator, Gripper, Hiking, Mystery, Parking et Zenotravel*

4.4.4 Évaluation de la longueur des plans

Les figures 4.11, 4.12 et 4.13 présentent les résultats de *GTOHP* et *SHOP* en termes de longueur de plans trouvés. L'axe des abscisses représente les problèmes de planification triés par ordre croissant en fonction du nombre d'objets et l'axe des ordonnées représente la longueur des plans en nombre d'actions. Les résultats de chaque planificateur sont représentés sur un tracé. Les résultats sont affichés dans la même échelle décimale de 0 à 180, excepté pour les domaines *Rover, Satellite et Barman* qui comportent des valeurs supérieures à 180 actions. Comme pour le critère de temps de traitement, les valeurs manquantes correspondent aux problèmes qui n'ont pas pu être résolus.

Ici la comparaison n'est pas réalisée dans le but de vérifier que *GTOHP* est plus efficace, mais plu-



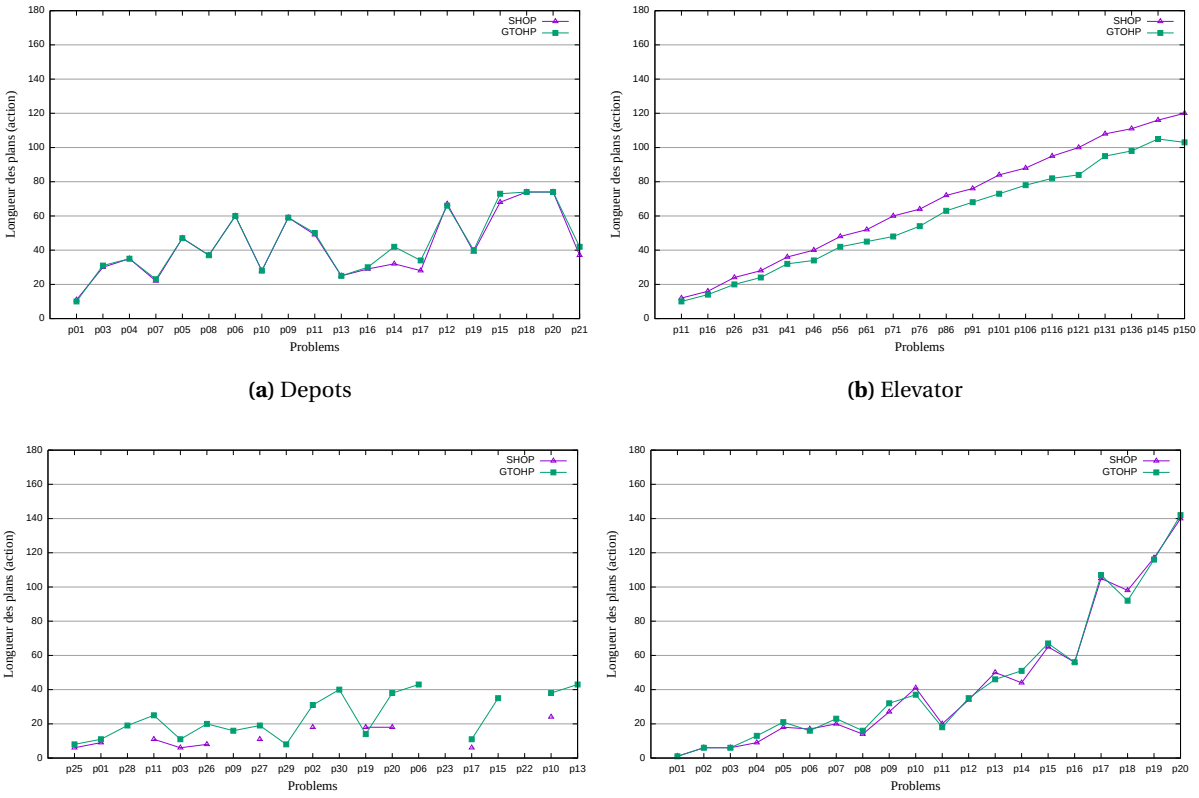
(a) Rover **(b)** Childsnack

(c) Satellite **(d)** Barman

FIGURE 4.11 – Graphiques représentant la longueur des premiers plans trouvé par les planificateurs *GTOHP* et *SHOP* pour les problèmes des domaines : (a) Rover, (b) Childsnack, (c) Satellite et (d) Barman

tôt pour vérifier que les plans qu'il génère restent similaires à ceux d'un planificateur HTN classique et que l'instanciation et la simplification n'introduisent pas de biais affectant négativement la longueur des plans. En HTN et en l'absence d'heuristique de recherche, cette dernière dépend en grande majorité de la définition du domaine, et plus précisément des méthodes de décomposition. Donc, avec une définition des domaines similaires, le résultat attendu doit être assez similaire pour les deux algorithmes.

Comme on peut le voir dans les figures 4.11, 4.12 et 4.13, les deux algorithmes génèrent des plans d'une longueur assez similaire, avec des longueurs parfaitement égales pour les problèmes des domaines *Childsnack, Blocksworld, Gripper et Hiking*. On observe aussi que la taille des plans augmente avec la taille des problèmes, à cause, notamment, de l'augmentation du nombre de tâches à réaliser. Cependant, la taille des plans n'est pas seulement liée au nombre de tâches à réaliser ou au nombre d'objets dans les problèmes, elle dépend aussi beaucoup de la complexité intrinsèque du problème



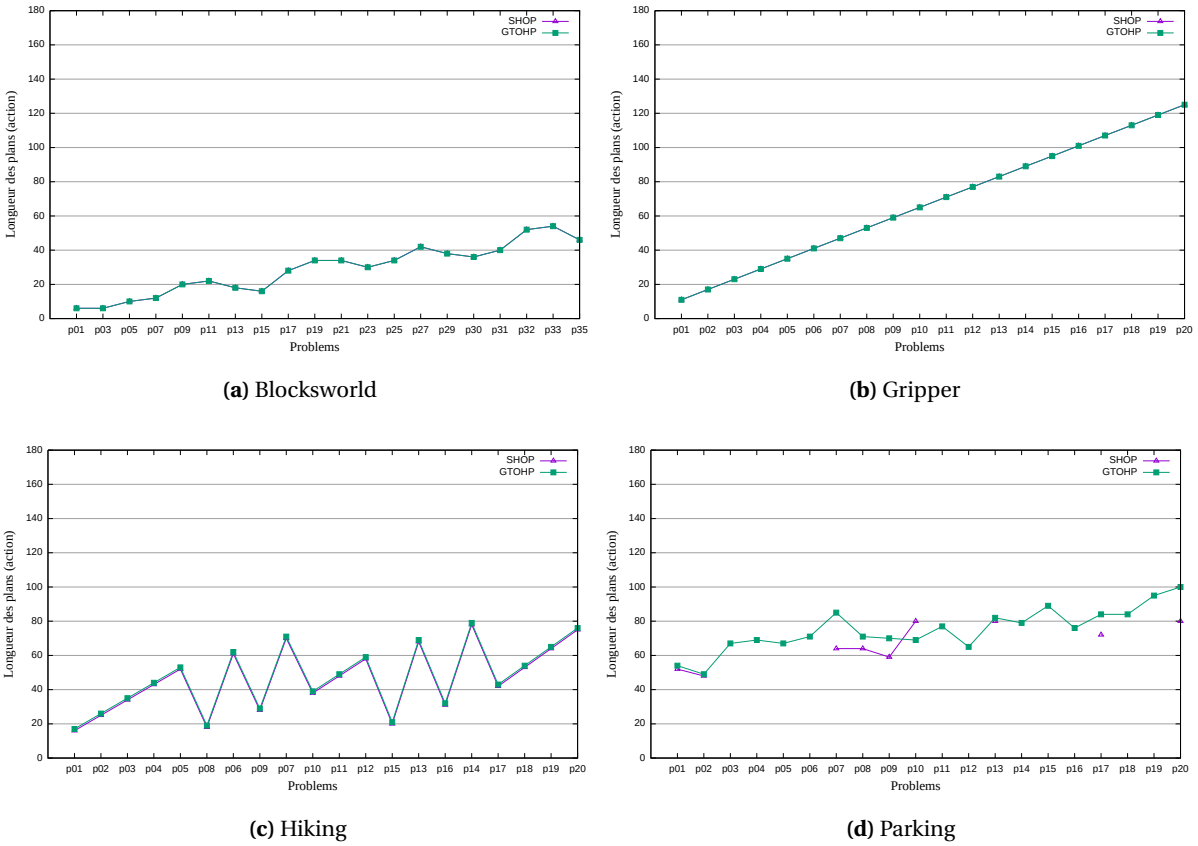
(a) Depots **(b)** Elevator

(c) Mystery **(d)** Zenotravel

FIGURE 4.12 – Graphiques représentant la longueur des premiers plans trouvé par les planificateurs *GTOHP* et*SHOP* pour les problèmes des domaines : (a) Depots, (b) Elevator, (c) Mystery et (d) Zenotravel

à planifier en termes d'état initial, des tâches à réaliser, ainsi que leur ordre d'exécution. Si on prend comme exemple la problématique de navigation du domaine *Rover*, le nombre d'actions à réaliser pour se déplacer d'un *waypoint* à un autre varie beaucoup en fonction de la position initiale, de la position finale et de la disposition de ces *waypoints*, même si leur nombre ne change pas. On peut aussi observer cela dans les résultats du domaine *Barman* où on a une différence de 33 actions entre les plans des problèmes 2 et 6 malgré le fait que les deux définissent 42 objets et 14 tâches à réaliser.

On observe dans les domaines *Mystery et Parking*, affichés respectivement dans les figures 4.12(c) et 4.13(d), que la longueur des plans trouvés par *GTOHP* pour les problèmes reste dans le même ordre de grandeur que les autres problèmes du même domaine. Cela signifie que l'incapacité de *SHOP* à trouver une solution n'est pas liée à la complexité du problème en terme de longueur de plan mais plutôt à sa complexité en terme de combinatoire. Cette combinatoire est mieux gérée par *GTOHP* grâce à l'instanciation et la simplification du problème et à la rapidité de décomposition et de vérifi-



(a) Blocksworld

(b) Gripper

(c) Hiking

(d) Parking

FIGURE 4.13 – Graphiques représentant la longueur des premiers plans trouvé par les planificateurs *GTOHP* et *SHOP* pour les problèmes des domaines : (a) Blocksworld, (b) Gripper, (c) Hiking et (d) Parking

cation des pré-conditions qui ne nécessitent plus d'étape d'instanciation et de simplification lors de la recherche.

Conclusion

Nous avons, dans ce chapitre, présenté un algorithme d'instanciation et de simplification des problèmes *HTN*, suivi d'un algorithme de planification *HTN* totalement ordonné et raisonnant sur des problèmes totalement instanciés et simplifiés. Les deux algorithmes ont été implémentés dans une extension de la librairie de planification *PDDL4J* Pellier & Fiorino (2018). Nous avons aussi démontré qu'à l'instar de l'instanciation des problèmes de planification classiques, celui des problèmes HTN permet aussi d'améliorer les performances des planificateurs *HTN*. De plus, cette étape de prétraitement des données offre la possibilité de réaliser des calculs d'atteignabilité qui peuvent encore

servir afin d'améliorer les performances des algorithmes de planification.

5

MIP : Un système de planification en mixed-initiative

Sommaire

5.1 Architecture du système 82
5.2 Fonctionnalités de lecture des connaissances 98
5.3 Fonctionnalités de modification des connaissances 106
5.4 Fonctionnalités de co-construction de plan 112
5.5 Évaluation du système MIP 117

Introduction

Faisant suite au développement d'un planificateur offrant une grande richesse de représentation et une grande capacité de décomposition du processus de planification, la deuxième partie de la thèse a été de construire un système en mixed-initiative qui organise l'interaction et les échanges entre l'utilisateur et le planificateur.

Nous présentons dans ce chapitre MIP : un système collaboratif de planification en mixed-initiative permettant la co-construction des plans pour l'aide à la configuration de systèmes complexes. Ce chapitre aborde l'architecture globale du système et présente en détails les interactions du système avec l'utilisateur, les échanges entre les modules du système, le fonctionnement de chaque module, ainsi que les fonctionnalités offertes. Le chapitre est organisé en deux parties, la première partie, sur l'architecture du système, présente les mécanismes qui se cachent derrière les interactions et la planification. La deuxième partie présente les fonctionnalités du système et leur utilité du point de vu de l'utilisateur.

5.1 Architecture du système

Nous avons implémenté l'architecture représentée par la figure 5.1 Dans le but de construire un planificateur en mixed-initiative adapté à l'assistance dans la configuration des systèmes complexes et répondant au critères d'évaluation exprimés dans le chapitre 2. Comme on peut le voir, cette architecture se décompose en quatre parties :

1. Le gestionnaire d'interaction, qui comme son nom l'indique, gère les interactions, en mixed-initiative, entre l'utilisateur et le système, ainsi que les échanges entre les modules ;
2. Les bases de connaissances, qui contiennent les connaissances sur le domaine et les problèmes de planification représentés avec différents niveaux de détails ;
3. Le module d'instanciation et de simplification qui réalise les opérations de calcul et de transformation des connaissances en effectuant le processus d'instanciation et de simplification présenté au chapitre 5.
4. Le module de planification qui s'occupe de la recherche de plans et de la production des informations nécessaires pour une interaction en mixed-initiative.

L'architecture du système se concentre sur les modules d'interaction, de planification, de représentation et de traitement des connaissances, sans détailler la ou les interfaces utilisées pour interagir avec l'utilisateur. Cela reflète l'objectif de cette thèse qui se concentre sur le développement des mé-

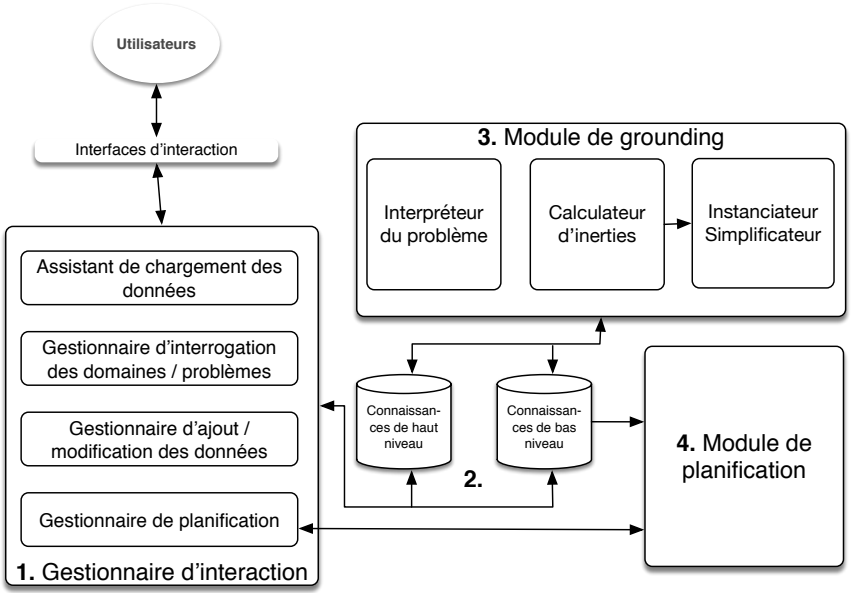


FIGURE 5.1 – Architecture du système de planification en mixed-initiative MIP.

canismes internes permettant une planification en mixed-initiative indépendamment de l'interface utilisée. Le développement d'une interface d'interaction adaptée pourra faire l'objet d'un nouveau sujet de recherche.

5.1.1 Gestionnaire d'interaction

Le gestionnaire d'interaction regroupe tous les modules qui prennent en charge les interactions en mixed-initiative avec l'utilisateur, pour cela :

- Ils reçoivent et interprètent les commandes de l'information, de modification, ou de planification qui sont exprimées par l'utilisateur en commandes internes du système ;
- Ils présentent les résultats et les réponses aux commandes de l'utilisateur ;
- Ils présentent les propositions de modification des connaissances du domaine ou du problème qui sont à l'initiative du système.

Le gestionnaire d'interaction s'occupe aussi de la gestion des échanges entre les modules et des appels de leurs fonctions afin de :

- Gérer l'affectation et l'ordonnancement des tâches nécessaires pour la résolution du problème ;
- Récupérer, manipuler et raisonner sur les données contenues dans les bases de connaissances ;
- Récupérer et formaliser les résultats des modules d'instanciation et de simplification ;
- Rassembler et interpréter les données produites pendant la recherche par le module de plani-

fication ;

— Récupérer et interpréter le résultat de la planification lorsqu'un plan solution est trouvé et même lors d'un échec.

En pratique, le gestionnaire de tâches traite et dispatche les requêtes en se fondant sur des commandes qui sont définies dans le système. Si par exemple l'utilisateur exprime une commande de planification cela passera par le mot clé « plan ». Il en est de même pour les autres commandes qui sont aussi définies par un mot clé. L'aspect modulaire du gestionnaire d'interaction permet de le faire évoluer et d'y inclure, par exemple, un module de reconnaissance de la parole qui induirait la requête et ces paramètres depuis une phrase en langage naturel.

Le gestionnaire d'interaction est composé de quatre sous-modules gérant chacun un ensemble de tâches précis. À chaque sollicitation du gestionnaire d'interaction, un ou plusieurs de ces modules sont appelés à répondre en fonction de la commande reçue. Cela peut aller du chargement des connaissances nécessaires à la planification, jusqu'au raisonnement sur les données pour expliquer le résultat d'une demande de planification, en passant par la modification du problème et la gestion des fonctionnalités liées à la planification, comme par exemple : la décomposition pas à pas ou l'incitation au changement, que nous détaillerons dans la deuxième partie de ce chapitre.

5.1.1.A Assistant de chargement des données

L'assistant de chargement des données permet au système MIP d'être « *domain-independant* ». Il permet de modéliser les propriétés et les états du domaine d'application, les connaissances sur les actions à réaliser, les tâches à accomplir, et de raisonner en se basant dessus, dans le but de résoudre les problèmes de planification liés à ce domaine. Par exemple, il peut être utilisé dans l'assistance de configuration de systèmes complexes, le pilotage de drones, ou tout autre activité nécessitant de générer automatiquement une suite d'actions permettant d'atteindre un but ou réaliser une tâche donnée. L'assistant de chargement interagit avec les utilisateurs afin de charger les données du domaine et du problème, sous forme de fichiers *HPDDL* contenant les définitions des prédicats, des opérateurs, des méthodes, ainsi que l'état initial [1] et le but à atteindre [2]. Avoir un module indépendant dédié au chargement des données permet une meilleure évolution et une adaptation plus facile au problème à gérer. Cela peut passer par la mise en place de procédures d'automatisation pour la récupération de l'état du système cible, ou encore l'ajout d'une interface pour récupérer les données du domaine et du problème sous une forme différente (*i.e.* depuis une ontologie en OWL-S).

1. L'état initial englobe généralement l'ensemble des propriétés représentant l'état du système cible au moment du chargement des données.

2. Le but à atteindre peut être un état du système cible, un ensemble de tâches à réaliser, ou la réalisation des deux.

5.1.1.B Gestionnaire d'interrogation

Le gestionnaire d'interrogation permet principalement de traiter les requêtes d'interrogation sur la base de connaissances. Il offre ainsi la possibilité à l'utilisateur de mieux comprendre le problème à travers l'interaction avec le système. Contrairement à une interface qui permet uniquement de visualiser l'immense quantité de données disponibles pour un problème, le gestionnaire d'interaction masque les données inutiles et présente à l'utilisateur les données les plus pertinentes pour la requête qu'il a exprimé. Pour cela, il interroge la base de connaissances sur deux niveaux d'abstraction, que nous aborderons plus en détails dans la section 5.2.1, et fournit ainsi les données pertinentes en fonction du niveau demandé de détails. Les requêtes que gère le gestionnaire d'interrogation ne se limitent pas seulement à la récupération de données, puisqu'il réalise aussi des raisonnements pour fournir des données liées les unes aux autres et dont le lien n'est pas forcément visible pour l'utilisateur. La figure 5.2 montre le fonctionnement du gestionnaire d'interrogation sur une requête d'interrogation nécessitant un raisonnement sur les données.

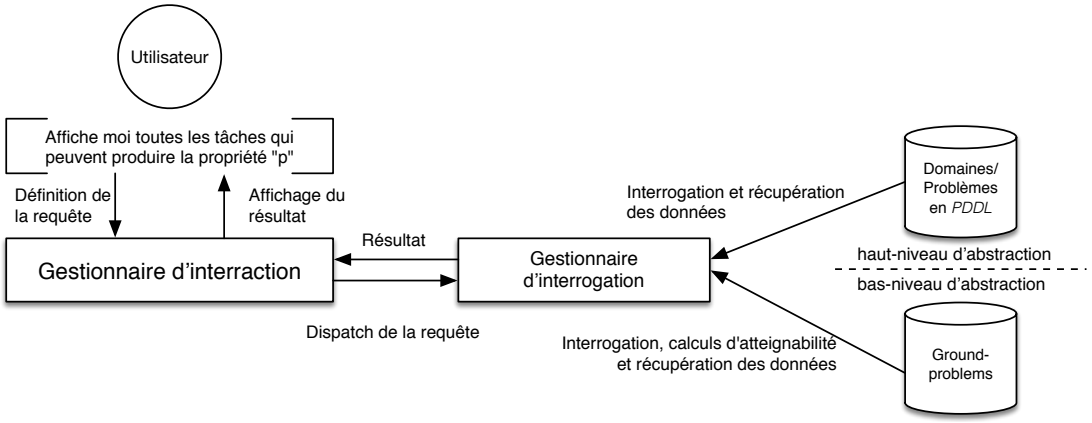


FIGURE 5.2 – Processus de gestion des requêtes d'interrogation.

Dans cet exemple, l'utilisateur demande au système de lui lister toutes les tâches du problème qui peuvent potentiellement produire le prédicat « p »[3]. La requête passe, en premier lieu, par le gestionnaire d'interaction qui la traite puis la dispatche vers le gestionnaire d'interrogation où le traitement approprié est exécuté. Ce dernier récupère alors les données et effectue les calculs d'atteignabilité nécessaires pour produire le résultat escompté. Enfin, le résultat de la requête est affiché à l'utilisateur après être passé par le gestionnaire d'interaction. Plusieurs types d'interrogations sont possibles à travers le système, elles seront présentées plus en détails dans la section 5.2

3. «p» est un prédicat du problème qui peut être un fait instancié ou un prédicat non instancié

5.1.1.C Gestionnaire de modifications

Le gestionnaire de modifications permet de traiter les requêtes d'ajout et de modification des données contenues dans la base de connaissances, au niveau des données de haut-niveau, stockées sous format *HPDDL*, et celui des données de bas-niveau, contenues dans des problèmes instanciés et simplifiés. Il fournit ainsi à l'utilisateur une façon plus simple d'ajouter ou de modifier des connaissances au problème. Il offre la possibilité de mieux cibler les données à modifier, en rendant leur recherche, leur récupération et leur modification, transparente pour l'utilisateur. Il permet aussi de gérer les messages d'erreurs pour s'assurer que les données ajoutées n'induisent pas d'inconsistances dans la base de connaissances, et cela sur les données de haut-niveau ainsi que sur les données de bas-niveau.

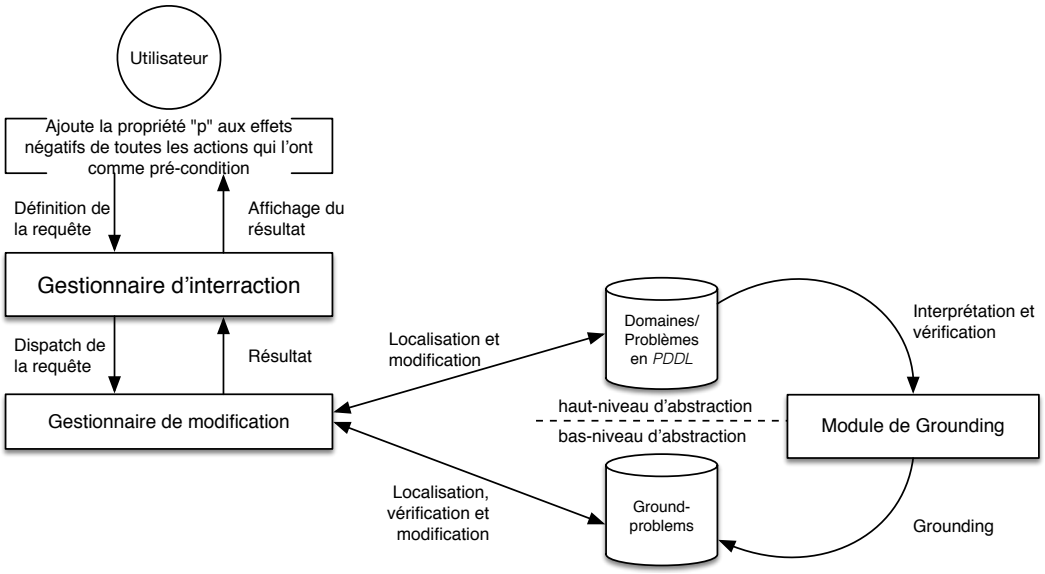


FIGURE 5.3 – Processus de gestion des requêtes de modification.

La figure 5.3 représente le processus induit par une requête de modification de la base de connaissances. On peut voir que le déroulement de ce processus reste très similaire à celui de la gestion des interrogations, en incluant en plus le module d'instanciation et de simplification. Si une modification est apportée aux données instanciées, à l'exception de l'état initial et des données modifiant les inerties, cela ne nécessite aucun traitement supplémentaire. Cette dernière sert principalement à apporter des modifications rapides sur les objets et à tester rapidement que la modification permet de résoudre un problème. Cependant, toute modification des données de haut niveau induit automatiquement une procédure d'instanciation et de simplification, afin de vérifier et générer le problème codé correspondant. Il existe plusieurs types de modifications, en fonction des structures affectées ou les données à modifier. Nous aborderons cela plus en détails dans la section 5.3

5.1.1.D Gestionnaire de planification

Le module de gestion de planification gère toutes les interactions traitant de la planification, qu'elles soient à l'initiative du module de planification ou de l'utilisateur. Il traite toutes les commandes de planification exprimées par l'utilisateur après qu'elles soient passées par le gestionnaire d'interaction. Il prend en charge, par exemple, les commandes de l'utilisateur visant à lancer une recherche de plan pour trouver une solution aux tâches exprimées dans le problème chargé ou à planifier, à la volée, une tâche définie dans la commande. L'utilisateur peut aussi interroger le planificateur sur le processus de planification en demandant des informations sur la tâche en cours de décomposition, le nombre de décompositions effectuées, la meilleure solution partielle trouvée jusqu'à présentent, etc. Il permet aussi de gérer les remontées du planificateur pour proposer une solution partielle, une contrainte à lever, ou encore pour signaler la découverte d'un meilleur plan solution. Tous ces retours apportent à l'utilisateur des informations précieuses qui l'aideront lors de la modification des données du domaine, le choix du plan à appliquer, ou le débogage si aucune solution n'est trouvée.

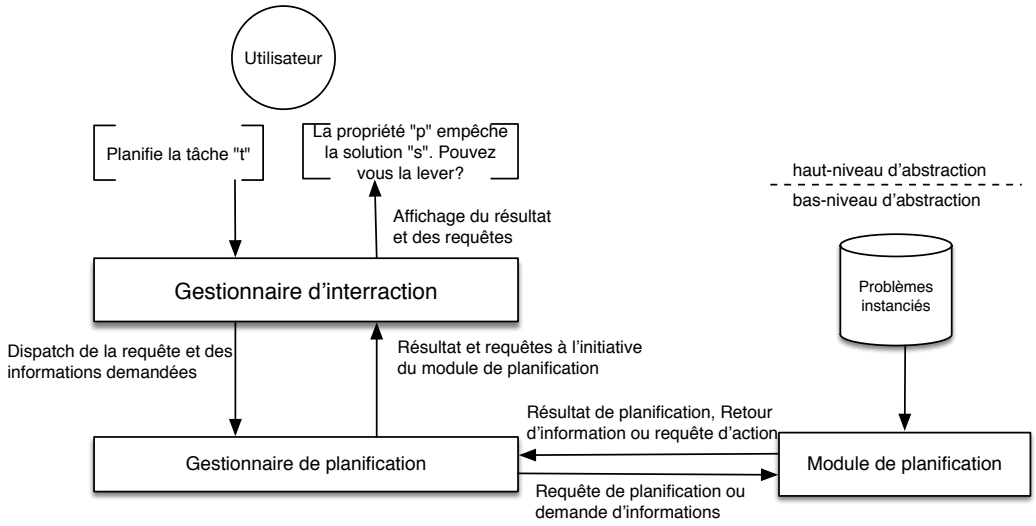


FIGURE 5.4 – Processus de gestion des requêtes de planification.

La figure 5.4 représente l'organisation des modules impliqués dans le processus de gestion de la planification. Ici on a un exemple de commande de planification exprimée par l'utilisateur, qui consiste à planifier une tâche à la volée [4]. La commande est prise en charge par le gestionnaire d'interaction qui la dispatche vers le gestionnaire de planification, qui va à son tour, apporter des mo-

4. La planification d'une ou plusieurs tâches à la volée se traduit par le remplacement des tâches contenues dans le task-network initial du problème codé par celles définies dans la commande de planification afin de lancer une nouvelle planification. Ces modifications ne sont pas sauvegardées dans le problème codé.

difications au problème codé, en fonction de la requête exprimée et ensuite faire appel au module de planification afin de trouver un plan solution au problème exprimé. Si une solution est trouvée, elle est remontée à l'utilisateur en suivant le chemin inverse. Si par contre aucune solution n'est trouvée, le module de planification fournit des informations sur la recherche, incluant des pistes pour expliquer cet échec. Ces informations sont traitées par le gestionnaire de planification et formalisées sous forme de proposition de modification du problème comme, par exemple, lever la contrainte qui bloque la solution.

5.1.2 Gestionnaire de base de connaissances

La base de connaissances contient toutes les structures et les données définies dans le domaine et le problème chargés lors de l'initiation de la recherche ainsi que celles résultants du processus d'instanciation et de simplification et modifiées au fil des interactions avec l'utilisateur. Elle se décompose en deux parties sur deux différents niveaux d'abstraction, stockant chacune les données du problème avec plus ou moins de détails.

5.1.2.A Avantages d'une représentation multi-niveaux des connaissances

Contrairement aux autres systèmes en mixed-initiative qui représentent les connaissances sous leurs forme brut, Le système MIP maintient deux niveaux d'abstractions avec des structures contenants des données de haut-niveau et des données de bas-niveau apportant ainsi plusieurs avantages :

— Offrir à l'utilisateur deux niveaux de détails sur une représentation des connaissances qui est déjà hiérarchique. Il peut ainsi manipuler des structures de haut niveaux ou des objets et des structures de bas-niveau. On abordera l'interaction sur deux niveaux plus en détails dans la section 5.2.1.

— Planifier avec des données instanciées permet de réaliser des étapes de planification plus performantes, comme le montre le chapitre 5.

— Les données déduites lors des études d'atteignabilité et stockées dans le problème instancié et simplifié fournissent à l'utilisateur des informations qu'il n'aurait pas pu calculer tout seul à partir du problème brut et offrent un moyen de répondre au critère de guidage qui est nécessaire dans un système en mixed-initiative. Ces données permettent aussi d'accélérer la planification, puisque elles ne sont pas calculés lors de la recherche et permettent d'élaguer les branches non pertinentes.

— Dupliquer les données avec deux niveaux d'abstractions permet de mieux gérer les modifications, puisque les petits changements de bas-niveau sont réalisés directement sur le problème

instancié alors que les modifications des données de haut niveau impliquent une étape de vérification et d'instanciation et simplification produisant une nouveau problème instancié et simplifié .

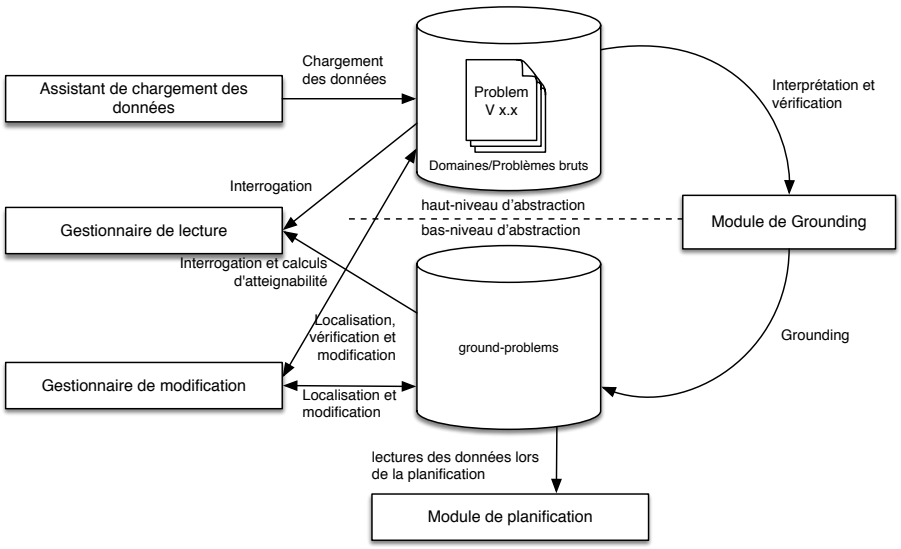


FIGURE 5.5 – Processus de gestion de la base de connaissances.

5.1.2.B Données de haut-niveau

Le premier niveau contient le domaine et le problème de planification exprimés sous format *HPDDL*, i.e. les prédicats, les types des objets, les constantes, les opérateurs, les méthodes, l'état initial, l'état but, ainsi que les tâches à planifier, chargés dans le système par l'assistant de chargement. Ces données évoluent au fil des interactions avec l'utilisateur, à travers les modifications qu'il apporte au domaine ou au problème. Un historique de versions est maintenu afin de permettre à l'utilisateur d'annuler les modifications apportées au cours de l'interaction. C'est très important lors d'une interaction en mixed-initiative, afin d'avoir plus de flexibilité et un meilleur guidage lors des modifications. Le niveau d'abstraction des données à ce niveau s'apparente à ce qui est utilisé dans d'autres systèmes en mixed-initiative (Myers et al. , 2002; Ferguson et al. , 1996; Immaneni & Cox, 2004) où les structures des domaines (*i.e*, les prédicats, les opérateurs et les méthodes) sont le plus souvent définies à l'aide de variables typées et utilisés dans l'état lors de la planification et l'interaction.

Dans ce niveau, l'utilisateur à accès, à travers le gestionnaire d'interaction, à des données de haut-niveau lui permettant d'avoir une bonne vision globale du problème. Soulignons que, chaque version du domaine et du problème est stockée sous forme de fichier HPDDL définissant les structures et les données utilisées. On peut imaginer, à l'avenir, l'interaction directe avec les objets *Java* générés par

l'interpréteur du problème, couplé avec une base de données et un système de gestion de versions avec arborescence afin de mieux gérer les modifications. Dans la figure 5.5, on peut voir les processus de lecture et d'écriture impliquant les données de haut-niveau de la base de connaissances. La première interaction se résume à l'écriture des données chargées par l'assistant de chargement, suivie directement par une étape de lecture pour la vérification et l'instanciation et simplification des données. Les données restent par la suite accessibles et modifiables, induisant automatiquement une étape d'interprétation, de vérification et d'instanciation et de simplification.

5.1.2.C Données de bas-niveau

Le deuxième niveau contient plusieurs versions qui regroupent les données instanciées ayant subi plusieurs traitements, lors de l'étape d'instanciation et de simplification, ainsi que les données calculées lors des études préalables d'atteignabilité. Cela regroupe entre autres, la liste des actions, des décompositions et des tâches pertinentes, ainsi que le task-network initial et les données sur l'atteignabilité. L'utilisateur a la possibilité d'interroger le système et d'effectuer des modifications sur un niveau d'abstraction très bas, où il peut directement manipuler les objets du problème et les structures pertinentes qui les utilisent. Cela lui offre la possibilité de mieux comprendre les détails du problème et d'y apporter des modifications rapides en manipulant les objets.

Les problèmes d'instanciation sont stockés dans la mémoire vive de la machine virtuelle *Java* permettant un accès et une modification très rapides. Un accès rapide au données est très important, surtout lors de l'étape de planification, qui implique énormément de lectures du problème. Comme on peut le voir dans la figure 5.5, les données instanciées et simplifiées sont accessibles en lecture par les gestionnaires de lecture et de modification ainsi que par le module de planification. Ce dernier implémente l'algorithme de planification *GTOHP* qui prend en entrée le problème instancié. Les données instanciées sont accessibles en écriture par les gestionnaires de modifications et le module d'instanciation qui écrit un nouveau problème ou remplace l'ancien par un nouveau après une modification des données de haut-niveau induisant une étape d'instanciation et de simplification.

5.1.3 Module d'instanciation et de simplification

Le module d'instanciation s'occupe du processus de pré-traitement des données, incluant : l'encodage, les vérifications syntaxiques, l'analyse sémantiques, le calcul des inerties, la simplification des données et l'instanciation du problème. Ces tâches sont réparties sur les trois sous modules qui constituent le module d'instanciation : L'interpréteur du problème s'occupe des tâches d'interpré-

tation, d'encodage et de vérifications ; Le calculateur d'inerties implémente les algorithmes de calcul d'inerties vus dans le chapitre 5 et réalise aussi les calculs d'atteignabilité ; L'instanciateur et le simplificateur implémente les algorithmes d'instanciation et de simplification présentés dans le chapitre 5. Le module d'instanciation intervient dans le processus de chargement initial des données et à chaque modification du problème brut, comme le montre les figures 5.3 et 5.5, afin de générer un nouveau problème encodé. La figure 5.6 montre les détails de l'implication des sous-modules d'instanciation et de simplification lors des processus de chargement des données et de modification.

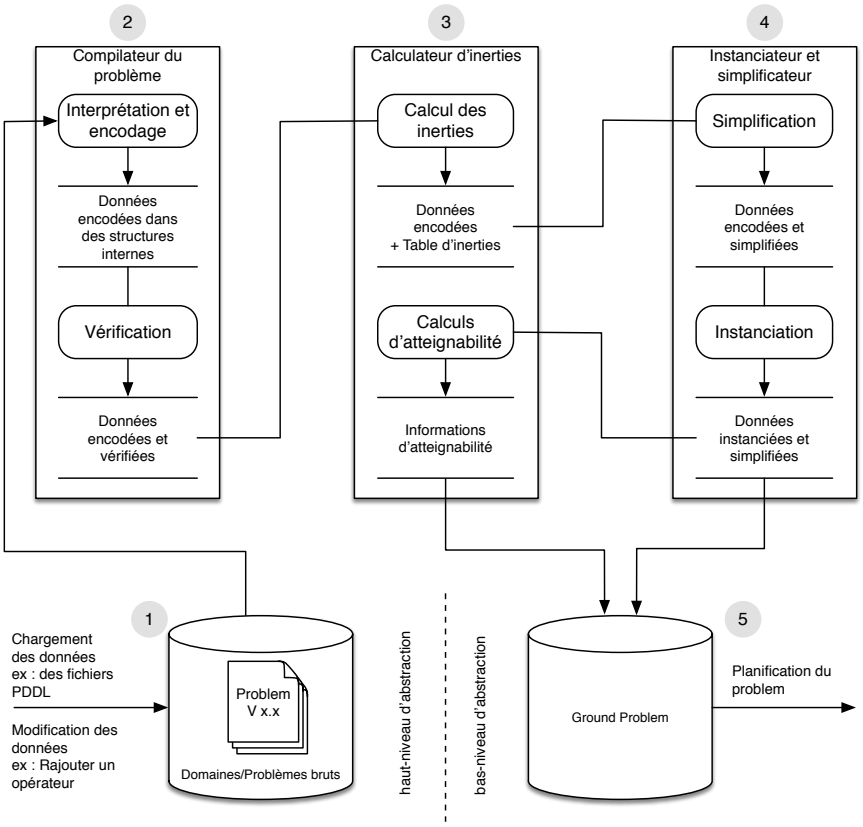


FIGURE 5.6 – Organisation des sous-modules impliqués dans le processus d'instanciation et de simplification.

5.1.3.A Compilateur du problème

Comme on peut le voir dans la figure 5.6, le compilateur du problème est le premier sous module impliqué dans le processus d'instanciation et de simplification. Les données brutes (Domaine + Problème) [5] sont récupérées depuis la base de connaissances sous forme textuelle pour être analysées et

5. Les connaissances brutes dans le système actuel sont représentées sous forme de fichiers de domaine et de problème PDDL. Cependant elle peuvent être modifiées pour s'adapter à la représentation des connaissances disponibles pour une

interprétées avec un analyseur lexical et syntaxique et encodées dans des structures internes. Cette représentation interne passe ensuite par une étape de vérification sémantique qui signale toute erreur ou incohérence dans les connaissances. L'interpréteur du problème fournit ainsi comme résultat un encodage interne et correcte des connaissances qui pourront être plus facilement utilisés par les autres modules.

5.1.3.B Calculateur d'inerties

Afin de calculer les inerties du problème, les connaissances encodées par l'interpréteur sont récupérées par le calculateur d'inerties qui recherche les propriétés concernées en suivant les algorithmes présentés dans le chapitre 5. Le résultat est donc une table qui montre pour chaque propriété du problème si c'est une inertie positive, négative, les deux, ou aucune des deux. Comme on a pu le voir précédemment, ces informations sont un pré-requis à l'étape de simplification du problème. Le calculateur d'inerties peut aussi effectuer des calculs d'atteignabilité afin d'ajouter dans le problème instancié des informations sur les propriétés et les tâches primitives atteignables à travers la décomposition d'une tâche composée. Ces informations peuvent s'avérer très précieuses pour le planificateur lors de la recherche ainsi que pour l'utilisateur afin de mieux comprendre le problème.

5.1.3.C Instanciateur et simplificateur

Après le calcul des inerties, l'instanciateur récupère les connaissances encodées et la table d'inerties afin de réaliser les normalisations, les simplifications ainsi que l'instanciation des opérateurs et des méthodes contenus dans les connaissances encodées en suivant les algorithmes d'instanciation et de simplification des opérateurs et des méthodes présentés dans le chapitre 5. Ensuite, les connaissances encodées et simplifiées sont enregistrées dans la base de connaissances sous forme de problème instancié contenant tous les faits, actions, tâches, décompositions du domaine qui se trouvent être pertinents pour ce problème. On note aussi qu'à la fin de l'étape d'instanciation et de simplification, les connaissances simplifiées sont utilisées pour le calcul d'atteignabilité dont le résultat est ajouté au problème.

application spécifique. Par conséquent, seuls les modules de chargement de données et l'interpréteur du problème vont changer.

5.1.4 Module de planification

Le module de planification est le cœur du système. Il effectue la recherche d'un plan solution pour un problème défini par l'utilisateur. Il utilise pour cela les connaissances du domaine contenues dans le problème. Afin de réaliser cette tâche, le module de planification implémente l'algorithme de planification *GTOHP* présenté dans la section 4.3. À la réception de la commande de planification, le module de planification récupère les données du domaine et de problème, puis lance la phase de recherche de plans solution, à travers la décomposition des tâches buts et la réécriture du task-network initial comme définit dans l'algorithme *GTOHP*.

En plus de l'implémentation de l'algorithme *GTOHP* pour la recherche, le module de planification implémente des mécanismes de méta analyse de la planification, d'analyse des solutions potentielles, de planification any-time, de comparaison de plans et d'interaction continue.

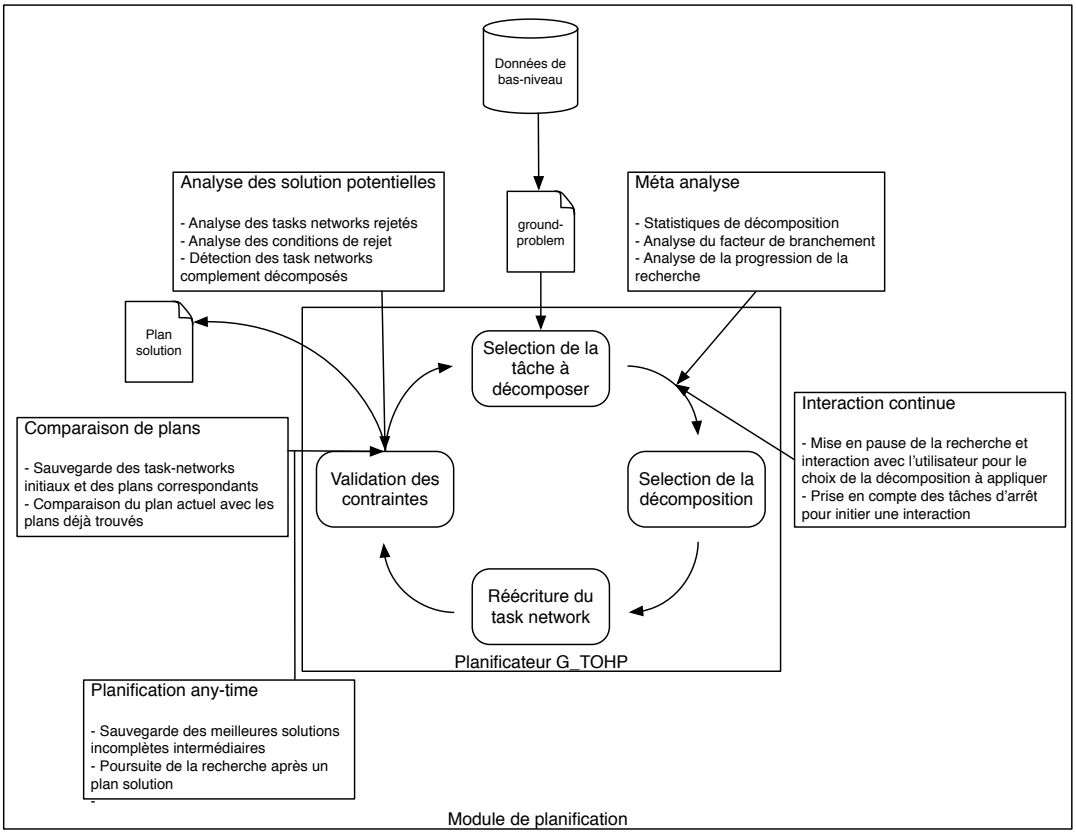


FIGURE 5.7 – Fonctionnement interne du module de planification.

La figure 5.7 montre l'organisation des tâches internes du module de planification. On a le planificateur *GTOHP* avec les quatre principales étapes d'un cycle de décomposition d'une tâche, à savoir : la sélection de tâche composée, la sélection de la décomposition, la réécriture du *Task-Network*, et la

validation des contraintes. Entres ces étapes, essentielles à la planification HTN, nous avons inséré les mécanismes nécessaires à l'analyse et la planification en mixed-initiative :

5.1.4.A Mécanisme de méta analyse de la planification

Ce mécanisme implémente plusieurs opérations de récolte de méta données sur le processus de recherche de plan. Elle est effectuée après l'étape de sélection de la tâche à décomposer afin de collecter les données sur le cycle de décomposition précédent et avoir assez d'informations sur la tâche à décomposer lors du cycle suivant. On peut lors de cette étape collecter des statistiques sur :

— Le nombre de décompositions réalisées pour estimer la complexité d'atteindre cette tâche ;
— Le facteur de branchement de chaque tâche pour estimer la complexité inhérente à sa réalisation ;
— L'état actuel de la recherche afin de donner la tâche en cours de calcul et donner à l'utilisateur une estimation de l'état actuel de la recherche ;
— La fréquence d'utilisation des décompositions afin de pointer les méthodes les plus importantes en terme de probabilité et de fréquence d'utilisation et fournit de ce fait à l'utilisateur des données lui permettant de réaliser des optimisations des connaissances ou des vérifications plus poussées ;
— Le nombre minimum de décomposition qui restent à réaliser avant de pouvoir atteindre une solution potentielle. Cela est possible grâce aux données sur les calculs d'atteignabilité réalisés par le module d'instanciation et de simplification et la représentation interne riche des task-networks.

Le planificateur *GTOHP* garde en mémoire toutes les feuilles de l'arbre de recherche afin de représenter toutes les possibilités de *backtracking*. Cela permet d'avoir toutes les solutions potentielles à un instant donné et d'avoir à la fois, les statistiques de recherche au niveau de chaque task-network et au niveau global de l'arbre de recherche.

5.1.4.B Mécanisme d'analyse des solutions potentielles

Après l'étape de validation des contraintes de l'algorithme *GTOHP*, si toutes les tâches du task-network en cours sont primitives et que toutes les contraintes sont satisfaites. Cela veut dire que le task-network représente un plan solution. Il est remonté à l'utilisateur en passant par les gestionnaires de planification et d'interaction pour la formulation d'une réponse compréhensive par l'utilisateur. Cependant, si le task-network contient des contraintes insatisfaites, il n'est pas directement supprimé, puisqu'il passe par l'étape d'analyse des solutions potentielles qui analyse les fac-

teurs de rejets des task networks qui peuvent être, en fonction de son état de décomposition, soit des contraintes qui n'ont pas pu être satisfaites, soit une tâche dont aucune des décompositions définies dans le domaine de connaissances n'est applicable dans l'état actuel du task-network[6]. Lorsque le task-network est primitif et complètement ordonné, les contraintes insatisfaites correspondent à un des deux cas suivants :

— Des pré-conditions d'actions qui n'ont pas pu être satisfaites dans les états précèdent leur application.
— Des faits de l'état but qui n'ont pas pu être atteints dans l'état final du task-network.

Le fait de remonter à l'utilisateur des task-networks primitifs dont les contraintes ne sont pas satisfaites permet de lui signaler des solutions incomplètes et de lui proposer une solution potentielle à condition qu'il lève les contraintes bloquantes. Cette fonctionnalité sera abordée plus en détails dans la section 5.4.1. Dans la même perspective, signaler une tâche dont aucune décomposition n'est applicable permet à l'utilisateur de voir rapidement les contraintes qui bloquent cette décomposition et de pouvoir les lever, cela lui permet aussi de définir des connaissances supplémentaires en réécrivant cette décomposition ou la méthode qui définit cet ensemble de décompositions.

Une utilisation très intéressante des données collectées en analysant les task-networks rejetés est faite dans la fonctionnalité d'explication de l'échec (section 5.4.6). En compilant ces données tout au long du processus de recherche, on a la possibilité de donner des statistiques sur le nombre de task-networks rejetées à cause d'une contrainte donnée, et de ce fait, faire sortir les contraintes qui n'ont jamais été vérifiées ou qui ont, statistiquement, fait rejeter le plus de solutions potentielles. Ces informations offrent à l'utilisateur une première piste de réflexion et lui permettent de trouver plus facilement l'erreur contenue dans les données du domaine ou du problème. Comme par exemple, voir facilement qu'un fait manque dans l'état initial du problème ou qu'un fait devait être produit par une action données mais qu'il manque dans ces effets positifs, etc.

5.1.4.C Mécanisme de planification any-time

On entend par planification any-time, la capacité du système, à chaque cycle du processus de recherche, de fournir à l'utilisateur la meilleure solution potentielle, ou de poursuivre la recherche de solution, même après en avoir trouvée une, dans l'espoir d'en trouver une meilleure. À la fin de la phase de validation des contraintes et d'analyse des solutions incomplètes, le module de planification réalise les deux opération suivantes :

6. L'état actuel du task-network est l'état calculé en appliquant, dans l'ordre, les transitions provoquées par les actions primitives ordonnées avant la tâche en cours de décomposition

— La sauvegarde des meilleures solutions incomplètes intermédiaires : Cela est réalisé lorsque le task-network n'est pas complètement décomposé et qu'il n'a pas été rejeté lors de l'étape de validation. Avant la fin de chaque cycle de décomposition d'une tâche dans un task-network, ce dernier est comparé à tous les task-networks actifs de l'arbre de recherche et celui qui est considéré comme le plus proche d'une solution est sauvegardé en tant que meilleur task-network. Afin d'évaluer ces task-networks, il faut d'abord estimer leurs distances du task-network représentant une solution. La définition d'un plan solution est donnée dans le chapitre 2.

Nous avons choisi de définir la distance entre un task-network et un task-network solution comme suit :

Définition 5.1. *La distance entre un task-network TN et son task-network solution TN_s est :*

$$\delta^{TN}_{TN_s} = |TN_{uconst}| + |TN_{ct}|$$

Où $|TN_{ct}|$ est le nombre de tâches composées du task-network et $|TN_{uconst}|$ est le nombre de contraintes non validées. Dans le cas d'une planification HTN totalement ordonnée, ce nombre correspond à la somme des pré-conditions contenus dans les actions et les faits du but non encore vérifiés dans l'état actuel du task-network.

Dans le cas d'une planification HTN partiellement ordonnée, ce nombre correspond au nombre des contraintes Before, After, Between et Protect qui n'ont pas encore été validées dans le task-network.

— La poursuite de la recherche après la présentation d'un plan solution : Dans le cas où le task-network en cours contient une solution, cette dernière est présentée à l'utilisateur comme plan solution. Cependant, le processus de recherche ne s'arrête pas pour autant, puisqu'il continue en arrière plan afin de trouver d'autres plans solution [7] qui seront présentés à l'utilisateur comme des alternatives au premier plan ou pour offrir plus de flexibilité d'exécution [8] de ces plans, s'ils sont très proches. En plus, toutes les solutions trouvées pourront être utilisées pour la comparaison de plans.

7. Trouver plusieurs plans solution signifie trouver plusieurs façons de réaliser la tâche définie par l'utilisateur

8. La flexibilité d'exécution se traduit par exemple par la possibilité de choisir entre deux actions ou réaliser une seule action du plan en choisissant entre deux objets

5.1.4.D Mécanisme de comparaison de plans

Les opérations nécessaires à la comparaison de plans sont réalisées après la validation du task-network en cours. Elles consistent à :

— Sauvegarder des task-networks initiaux et des plans correspondants dès qu'une première solution est trouvée. En plus de sauvegarder les plans trouvés pour les comparer avec d'autres plans solution obtenus grâce au mécanisme de planification any-time, cela permettrait de faire le lien entre l'état initial, le but, les tâches but et le plan solution trouvé. Avec des mécanismes de recherche spécifiques, on pourrait, à chaque nouvelle commande de planification, effectuer une simple recherche afin de vérifier qu'une solution n'a pas déjà été trouvée pour le problème exprimé. Cependant, en sachant que les connaissances du domaines peuvent avoir été modifiées entre le moment où la solution a été trouvées et celui ou le problème a été posé, une validation de ce plan solution avec les connaissances à jour est nécessaire. Cette opération est réalisée, en vérifiant que toutes les actions sont atteignables depuis les tâches à réaliser et en appliquant les actions du plan à partir de l'état initial jusqu'au but en validant les contraintes.

— Comparer le plan actuel avec les plans déjà trouvés. Pour cela, un ou plusieurs critères d'évaluation sont pris en compte pour calculer le coût global du plan. Le cas le plus simple est de considérer la longueur du plan comme critère d'évaluation : plus le plan est court, plus la solution est considérée comme performante. Ce mécanisme fournit la base de comparaison utilisée dans la fonctionnalité d'amélioration des plans (section 5.4.4).

5.1.4.E Mécanisme d'interaction continue

On entend par interaction continue, la capacité du système, à chaque cycle du processus de recherche, à interagir avec l'utilisateur afin de prendre en compte son choix dans la sélection de la décomposition à appliquer ou pour analyser le contenu du task-network à ce niveau de décomposition. Permettre à l'utilisateur de choisir la décomposition à appliquer revient à lui donner la possibilité de choisir la façon dont la tâche est réalisée. Cela lui permet aussi de suivre pas à pas le raisonnement du planificateur, à travers la fonctionnalité de décomposition pas à pas (section 5.4.2), afin de mieux comprendre le processus qui a mené à une solution donnée ou de pouvoir mieux rectifier les erreurs qui mènent à un échec. Avec une planification HTN totalement ordonnée, l'intervention de l'utilisateur n'est possible qu'après le choix de la tâche à décomposer, mais il est possible aussi de permettre à l'utilisateur de choisir la tâche à décomposer avec une planification HTN partiellement ordonnée, apportant plus de flexibilité et d'adaptabilité au système.

Le mécanisme d'interaction continue n'est pas activé par défaut, il doit être demandée explicitement par l'utilisateur dans la commande de planification où il a aussi la possibilité de définir le niveau d'interaction : lors de tous les choix de décomposition, toutes les n décompositions, ou sur une tâche t donnée. Pour les deux derniers choix, l'utilisateur doit être capable de suivre les décompositions dans plusieurs task-networks, d'où la nécessité de lui afficher toutes les informations du task-network et les opérations effectuées, *i.e.* le choix task-network en cours, la tâche en cours de décomposition, la liste des tâches et des contraintes du task-network, etc. Ces informations auront déjà été collectées lors de l'analyse des solutions et de la méta analyse. Toutes les opérations ajoutées pour la mixed-initiative influent sur les performances de la planification en termes de temps de recherche. Cependant, avec la représentation totalement instanciée du problème et des connaissances du domaine, les opérations réalisées sont beaucoup moins coûteuses en mémoire et en temps de calcul et peuvent être désactivées pour revenir aux performances initiales de *GTOHP*.

5.2 Fonctionnalités de lecture des connaissances

La compréhension, l'acceptabilité et la facilité de prise en main sont des enjeux majeurs pour les systèmes en mixed-initiative. Après avoir mentionné quelques unes des fonctionnalités du système lors de la présentation de son architecture et de ces modules, nous allons aborder en détails, dans cette section, les fonctionnalités de lecture et de compréhension des connaissances ainsi que leurs prise en main par l'utilisateur. Le but recherché à travers ces fonctionnalités de prise en main est de fournir à l'utilisateur le maximum d'informations structurées et ciblées, indépendamment des interfaces homme-machine utilisées, afin de mieux le guider et faciliter sa compréhension des connaissances stockées.

5.2.1 Interaction sur deux niveaux d'instanciation et de simplification

Le point commun entre toutes les fonctionnalités du système MIP est la possibilité offerte à l'utilisateur de pouvoir interagir avec le système, soit au niveau des structures de définition des connaissances de haut niveau, à savoir : les prédicats, les opérateurs, et les méthodes qui comportent des variables typées, soit au niveau des structures de bas niveau comme : les faits, les actions, et les décompositions qui sont contenues dans le problème instancié et simplifié.

Les structures de bas niveau sont calculées à partir de celles du haut niveau lors du processus d'instanciation et de simplification. Toute modification dans les données de haut niveau affecte forcément celles de bas niveau et nécessite une nouvelle instanciation et simplification. Cependant, l'in-

verse n'est pas forcément vrai, puisque la modification d'une structure de bas niveau avec un groupe d'objets, comme une action sur un objet *"a"* par exemple, n'affecte pas forcément toutes les autres structures avec d'autres objets, comme une autre action sur l'objet *"b"* par exemple.

La modification des connaissances de haut niveau permet alors de réaliser de réelles modifications sur le domaine et le problème et affecte ainsi durablement, et parfois de façon très conséquente le problème codé. Par contre, la modification des connaissances de bas niveau sert essentiellement à effectuer des modifications locales sur les objets afin de tester rapidement le résultat d'une modification ou de reporter un comportement particulier sur un objet qui pourra être rajouté dans les connaissances de haut niveau sous forme de contrainte particulière sur ce dernier.

On peut voir dans le code ci-dessous un exemple de représentation textuelle de la méthode de navigation du domaine *Rover* sur deux niveaux d'abstraction affichés ensembles. Dans le premier niveau, les paramètres de la méthode sont des variables typées et la méthode en elle même définit un squelette de la façon dont un objet de type *ROVER* doit naviguer pour atteindre un objet de type *waypoint*. Dans le deuxième niveau, on trouve une partie des décompositions qui représentent la méthode de haut-niveau. Leurs paramètres sont des objets et ils définissent la procédure exacte à suivre afin que le *rover0* navigue vers un *waypoint* précis. L'utilisateur a ainsi la possibilité de travailler sur la méthode globale ou sur les décompositions possibles dans le problème actuel, ces décompositions sont celle qui n'ont pas été simplifiées lors de l'instanciation et la simplification. Le système réduit ainsi la complexité du problème et permet, à travers la représentation de bas-niveau, à l'utilisateur d'accéder directement aux décompositions pertinentes.

```
Haut-niveau d'abstraction : La methode
(:method do_navigate
    :parameters (?x - rover ?to - waypoint)
    :expansion  (
                 (tag t1 (visit ?from))
          (tag t2 (do_navigate ?x ?from ?to))
          (tag t3 (unvisit ?from))
         )
    :constraints(and (before (at ?x ?from) t1))
)
```

```
Bas-niveau d'abstraction : Les decompositions
(:method do_navigate
 :parameters (rover0 waypoint1)
```

```
 :expansion ((tag t1 (visit waypoint2))
       (tag t2 (do_navigate rover0 waypoint2 waypoint1))
       (tag t3 (unvisit waypoint2)))
    :constraints(and (before (at rover0 waypoint2) t1)))

(:method do_navigate
 :parameters (rover0 waypoint1)
 :expansion ((tag t1 (visit waypoint3))
       (tag t2 (do_navigate rover0 waypoint3 waypoint1))
       (tag t3 (unvisit waypoint3)))
    :constraints(and (before (at rover0 waypoint3) t1)))
```

La représentation des connaissances sur deux niveaux d'abstraction permet aussi de fournir à l'utilisateur des connaissances directes qu'il aurait dû déduire seul à partir de la définition du domaine. On peut citer, comme exemple, les types de variables non déclarées, comme la variable *?from* de l'exemple précédent, dont le type n'est pas définit dans la méthode *do_navigate* et qui nécessiterait que l'utilisateur trouve les contraintes et les tâches de la méthode qui contient cette variable puis fasse le lien avec leurs définitions afin de déduire son type, tout en vérifiant que ces types sont compatibles et de trouver le plus restrictif d'entre eux. Au lieu de cela, l'utilisateur peu accéder directement à une des décompositions définies par la méthode *do_navigate* et voir l'objet correspondant à la variable en question et d'afficher, ensuite, son type.

5.2.2 Le listing des structures du problème

Le listing des structures est la première fonctionnalité de prise en main du problème. Elle est prise en charge par le gestionnaire d'interrogation et définit l'ensemble des commandes permettant de lister les structures définies dans la base de connaissances sur les deux niveaux d'abstraction. Le premier objectif du listing est de fournir à l'utilisateur une interface de lecture du problème qui s'adapte à son niveau et qui permet de réduire sa charge de travail en décomposant l'affichage du problème et en lui évitant d'aller lire le fichier de définition du domaine en HPDDL, ce qui peut être très compliqué, surtout pour les utilisateurs qui ne sont pas familiers avec le langage HPDDL. Les principales commandes de listing possibles à travers le système MIP sont illustrées dans le tableau 5.1.

Le système fournit, à travers cette fonctionnalité, la liste des structures définies dans le domaine,

Commandes	**Objectif**
List operators	Affiche tous les opérateurs définis dans le domaine
List actions	Affiche toutes les actions pertinentes du coded-problem
List methods	Affiche toutes les méthodes définies dans le domaine
List decompositions	Affiche toutes les décompositions pertinentes du coded-problem
List tasks	Affiche toutes les tâches pertinentes contenues dans le coded-problem
List primitive tasks	Affiche toutes les tâches primitives du coded-problem
List composed tasks	Affiche toutes les tâches composées du coded-problem
List types	Affiche tous les types d'objets définis dans le domaine
List objetcs	Affiche tous les objets définis dans le domaine et leurs types
List predicates	Affiche tous les prédicats définis dans le domaine
List facts	Affiche tous les faits pertinents contenus dans le coded-problem
List goal tasks	Affiche les tâches but définies dans le problème ou par l'utilisateur
List initial state	Affiche l'état initial du problème
List goal state	Affiche l'état but du problème
List inertia	Affiche les prédicats et les faits qui sont des inerties dans le problème
List irrelevant actions	Affiche toutes les actions qui ont été supprimées du coded-problem
List irrelevant decomps	Affiche toutes les décompositions qui ont été supprimées du coded-problem

TABLE 5.1 – Ensemble des commandes permettant le listing des structures de la base de connaissance

comme les opérateurs, les méthodes, les prédicats, ainsi que les objets et leurs types respectifs. En ce qui concerne le problème, le système permet de lister les objets qui y sont définis, les tâches but, l'état initial, et l'état but. On peut voir aussi que le système offre à l'utilisateur une interface d'interrogation non-seulement sur les structures du domaine et du problème, mais aussi sur celles calculées lors de l'insanciation et la simplification, comme les actions, les décompositions, les faits, les inerties ou même les compositions supprimées.

Lors de la réception de la commande de listing, le système accède à la base de connaissances et charge la version courante des structures demandées. Comme dans la figure 5.8, si l'utilisateur demande de lui lister tous les objets du domaine, ces derniers ne correspondront pas forcément aux objets définies dans les fichiers *HPDDL* chargés initialement par le gestionnaire de chargement, mais

plutôt à ceux contenus dans la version courante du domaine. En effet, une nouvelle version du domaine et du problème est créée à chaque modification effectuée par l'utilisateur nécessitant une phase d'instanciation et de simplification, la dernière version obtenue est alors considérée comme la version courante.

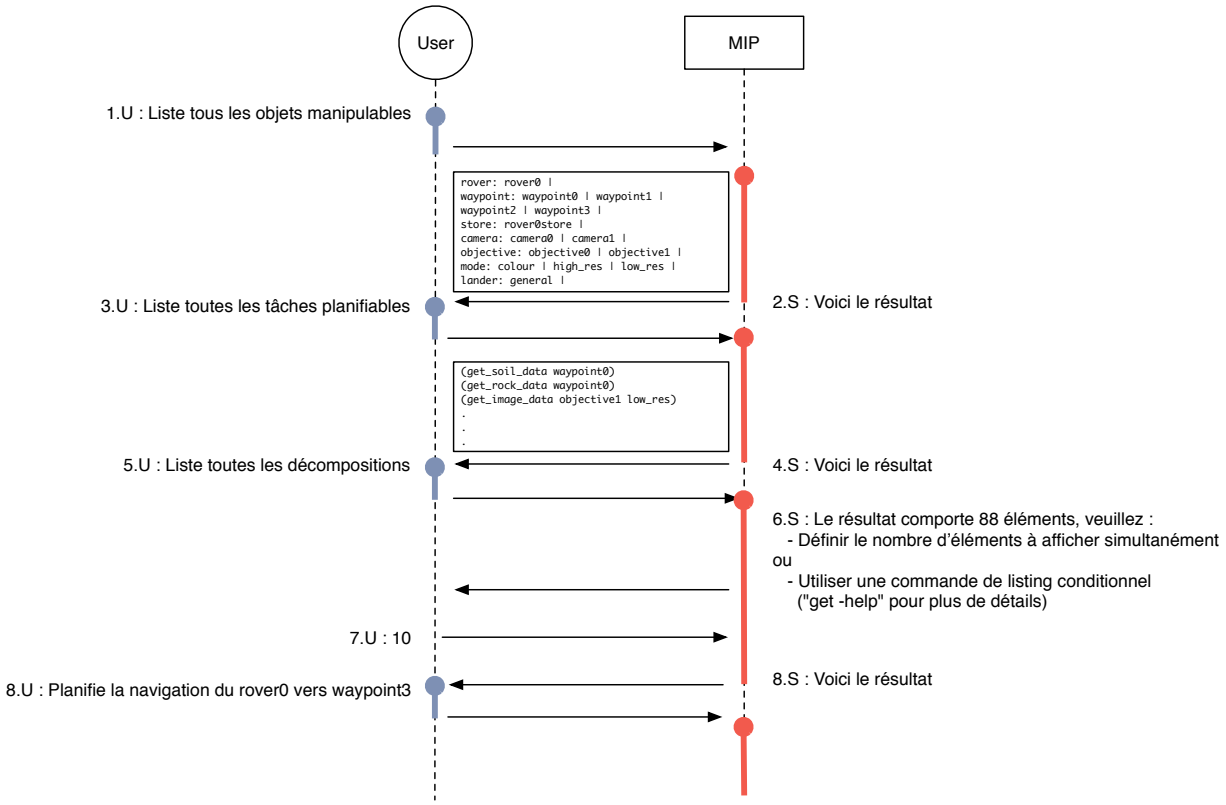


FIGURE 5.8 – Exemple typique d'échange utilisant la fonctionnalité de listing.

La figure 5.8 représente un exemple d'échange entre l'utilisateur et le système utilisant la fonctionnalité de listing sous forme d'un diagramme de séquences projetant les initiatives prises par chaque partie. L'utilisateur explore les connaissances du domaine et du problème en exprimant des requêtes de listing qui sont satisfaites par le système, à l'exception de l'échange 6 où le système demande à l'utilisateur de faire un choix entre définir un plafond d'affichage ou d'exprimer une requête de listing conditionnel. Après avoir obtenu les informations désirées, l'utilisateur a décidé de lancer la planification d'une nouvelle tâche utilisant ainsi une autre fonctionnalité du système que nous verrons dans les prochaines sections.

On peut voir à travers cet exemple que la réponse du système aux commandes de listing se traduit toujours par l'affichage du résultat correspondant. Cependant, dans la majorité des cas, les domaines définis sont très longs et les problèmes exprimés sont complexes, ce qui donne une liste de résultats

extrêmement longue. Le système propose alors à l'utilisateur un affichage progressif du résultat ou de formuler une commande de listing conditionnel avec, par exemple, le nom de la structure recherchée ou les propriétés atteignables à travers elle. Ce genre de commande est appelé listing conditionnel et sera présenté en détails dans la prochaine section.

5.2.3 Le getting des structures du problème

La fonctionnalité de getting des structures du problème est un listing particulier qui rajoute une ou plusieurs conditions sur les structures demandées. Comme le listing, le getting est pris en charge par le gestionnaire d'interrogation et permet à l'utilisateur d'avoir des informations sur le domaine et le problème sur les deux niveaux d'abstraction. L'objectif principale de cette fonctionnalité est de fournir des informations très spécifiques, plus ciblées que celles obtenues à travers le listing, et surtout, réaliser des calculs que l'utilisateur est incapable de reproduire sans assistance afin de faire le lien et de regrouper des structures en fonction d'un critère donné. On peut citer comme critère de regroupement : Le fait produit, la sous-tâche atteignable, ou encore, la pré-condition nécessaire à l'exécution d'une action ou l'application d'une décomposition. Les principales commandes de getting possibles à travers le système MIP sont illustrées dans le tableau 5.2.

On peut diviser ces commandes de getting en deux catégories :

— Les commandes de getting en fonction de la signature : Cette catégorie contient les commandes de 1 à 6 du tableau, qui regroupent les structures demandées en fonction du nom ou de leurs signatures [9]. Pour les commandes de getting sur des objets, i.t. 5 et 6, on s'intéresse à l'objet concerné par le fait, l'action ou la décomposition, en affichant par exemple toutes les actions qui concernent l'objet *waypoint0*. La réponse du système à ce genre de commande ne nécessite pas de réaliser des calculs supplémentaires, puisqu'il faut simplement explorer les données stockées dans la base de connaissances et extraire celles correspondant au nom ou à la signature.

— Les commandes de getting en fonction de l'application : Les commandes de cette catégorie, i.e. de 7 à 11, regroupent les structures en fonction de la condition de leur application ou du résultat obtenu après leur application. Il est très intéressant de voir les opérateurs qui produisent un fait donné ou qui ont besoin que ce fait soit vrai pour être appliqués. Cela permet, notamment, à l'utilisateur de voir pourquoi ce fait n'a pas été produit, ou pourquoi une action donnée n'a pas été appliquée. Cet outil offre à l'utilisateur la possibilité de construire un

9. La signature d'une structure est composée du nom de cette structure et de ces paramètres. Par exemple, la signature de la première décomposition *do_navigate* est *do_navigate rover0 waypoint1*. Par contre, la signature d'une méthode ou d'un opérateur doit être spécifiée avec les types des paramètres, comme ce-ci : *do_navigate ROVER WAYPOINT*

Commandes	**Objectif**
Get [operators | methods] with [opt]	Affiche les [opérateurs | méthodes] dont le nom ou la signature correspond à [opt]
Get [actions | decompositions] with [opt]	Affiche les [actions | décompositions] dont le nom ou la signature correspond à [opt]
Get tasks with [opt]	Affiche les tâches dont le nom ou la signature correspond à [opt]
Get predicates on [type]	Affiche les prédicats dont un des arguments est du type [type]
Get facts on [obj]	Affiche les faits dont un des paramètres est [obj]
Get [actions | decompositions] on object [obj]	Affiche les [actions | décompositions] dont un des paramètres est [obj]
Get actions [producing | consuming] [fact]	Affiche les actions qui [produisent | consomment] le fait [fact]
Get decompositions [producing | consuming] [fact]	Affiche les décompositions qui peuvent [produire | consommer] le fait [fact]
Get [actions | decompositions] needing [fact]	Affiche les actions qui ont dans leurs pré-conditions le fait [fact]
Get task [producing | consuming] [fact]	Affiche les tâches qui [produisent | consomme] le fait [fact]
Get composed task that may genarate [task]	Affiche les tâches composées qui peuvent générer la sous-tâche [task]

TABLE 5.2 – Ensemble des commandes permettant le getting des structures de la base de connaissance

schéma de compréhension liant les structures entre elles et de détecter rapidement une erreur de définition ou un manque d'information.

La figure 5.9 montre un exemple d'utilisation de la fonctionnalité de getting sur le domaine *rover* pour vérifier qu'elle est la tâche à effectuer afin de transmettre les données des échantillons de sol du *waypoint0*. Dans cet exemple, l'utilisateur commence d'abord par vérifier que le but contient bien le fait *communicated_soil_data waypoint0* en affichant le but avec une commande de listing. Après cela, il demande qu'elles sont les actions qui produisent ce fait pour comprendre le mécanisme de transmission et les objets impliqués dans le processus. Sans connaître la disposition des *waypoint0* et la position du *rover*, l'utilisateur est incapable de choisir l'action adéquate, il demande alors les tâches

composées qui pourraient produire le fait désiré, puis, affiche celles qu'il a choisit pour comprendre son fonctionnement. À ce moment, il peut voir déjà qu'il n'y a qu'une seule façon de faire avec le *rover0* sans prendre en compte son point de départ. Pour finir, il lance la planification de cette tâche et obtient un plan avec le *rover0* commençant au *waypoint1*. Il peut même vérifier que *rover0* est bien dans le *waypoint1* dans l'état initial en l'affichant.

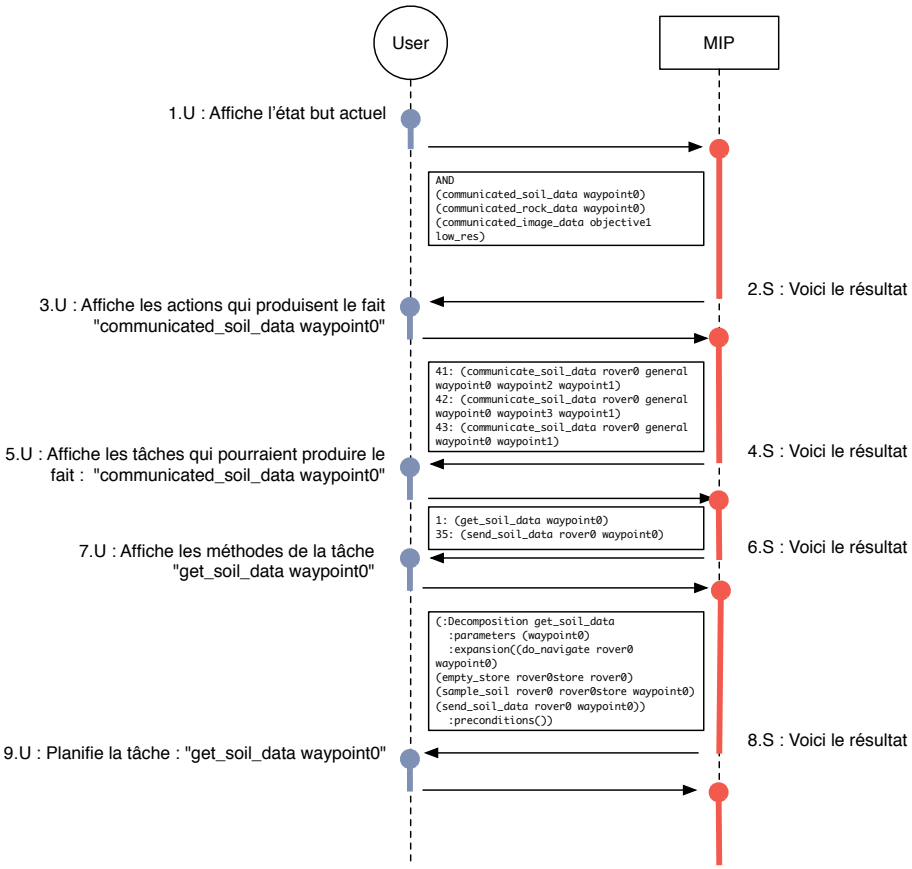


FIGURE 5.9 – Exemple d'échange utilisant la fonctionnalité de getting.

La compréhension des informations stockées dans la base de connaissances et celles du problème à résoudre est aussi importante, peut être même plus importante, que la compréhension des mécanismes de planification. Maîtriser une problématique de planification, même simplifiée, comme *rover* reste une tâche très délicate, surtout pour des utilisateurs non experts dans le domaine d'application ou dans la planification automatique. Nous avons présenté dans cette section les fonctionnalités du système MIP permettant de décomposer le problème et les connaissances du domaine pour une meilleure gestion de la charge de travail. Ces fonctionnalités permettent aussi d'effectuer des raisonnements sur les structures définies dans le domaine ou calculées lors de l'instanciation et de la simplification afin de regrouper les éléments sur un critère donné et aider l'utilisateur dans

son raisonnement. La lecture des connaissances à travers l'interface du système peut s'avérer très intéressante pendant la recherche d'erreurs et de deboggage, ou lors de leurs modifications avec les fonctionnalités offertes par le système MIP présentées dans la section suivante.

5.3 Fonctionnalités de modification des connaissances

Afin de répondre au critère de guidage de l'utilisateur et de contrôle, ce dernier doit avoir le contrôle sur des données stockées dans la base de connaissances, tout en étant guidé lors des opérations de modification des données. Il est nécessaire de donner la possibilité à l'utilisateur de modifier et de modeler ces données selon son mode de raisonnement, tout en s'assurant de leurs cohérence, en gardant l'historique des modifications et en sauvegardant les versions précédentes pour pouvoir revenir en arrière en cas d'erreur. Les fonctionnalités de modification des connaissances présentées dans cette section permettent de répondre à cette problématique à travers des commandes de modification ciblées, i.e. qui ne donnent pas accès à l'ensemble des connaissances en même temps et qui ne permettent pas de réaliser plusieurs modifications à différents niveaux sans vérifications intermédiaires. Ces modifications peuvent intervenir sur les deux niveaux d'abstraction et déclencher automatiquement les processus de vérification de la cohérence et d'instanciation et simplification nécessaires.

5.3.1 Ajout, suppression et mise à jour des connaissances

Les opérations d'ajout, de suppression et de mise à jour peuvent concerner toutes les structures du domaine stockées dans la base de connaissances. Elles sont réalisées sur des structures de haut-niveau pour une modification en profondeur induisant un nouvelle instanciation et simplification, et sur les structures de bas-niveau pour faire des tests de modification précis et restreint qui pourront être généralisés avec des modifications de haut-niveau. Ces opérations peuvent être réalisées pour mettre à niveau les données de la base de connaissances, (*e.g.*, ajouter une nouvelle tâche, action ou décomposition, modifier un type, etc.), et peuvent aussi intervenir dans plusieurs étapes de l'interaction en mixed-initiative, (*e.g.*, après la proposition du système de lever une contrainte sur l'application d'une action ou d'une décomposition, rajouter une nouvelle décomposition à la volée, etc.). Le tableau 5.3 regroupe l'ensemble des instructions de modification possibles à travers le système MIP.

L'ajout ou la suppression d'une structure de haut-niveau, *e.g.*, les opérateurs ou les méthodes a pour but de mettre à jour les connaissances fondamentales du domaine de façon importante et doit être réalisée par un expert du domaine d'application. Cela va entraîner l'ajout ou la suppression d'un

Commandes	**Objectif**
[Add | Remove | Modify] the operator [signature]	[Ajoute |Supprime| Modifie] l'opérateur correspondant à [signature]
[Add | Remove | Modify] the method [signature]	[Ajoute |Supprime| Modifie] la méthode correspondant à [signature]
[Add | Remove | Modify] the action [signature]	[Ajoute |Supprime| Modifie] l'action correspondant à [signature]
[Add | Remove | Modify] the decomp [signature]	[Ajoute |Supprime| Modifie] la décomposition correspondant à [signature]
[Add | Remove] the type [type]	[Ajoute |Supprime] le type [type]
[Add | Remove | Modify] the predicate [signature]	[Ajoute |Supprime| Modifie] le prédicat correspondant à [signature]
[Add | Remove] the object [obj]	[Ajoute |Supprime] l'objet [obj]

TABLE 5.3 – Ensemble des commandes de modification des structures de la base de connaissance

type de comportement particulier, *e.g.*, une famille d'actions sur des objets d'un certain type ou une méthode à suivre pour réaliser un certain type de tâches. Comme mentionné à plusieurs reprises, ce type de changement induit automatiquement une nouvelle étape d'instanciation et de simplification et une génération d'un nouveau problème codé.

L'ajout ou la suppression d'une structure de bas-niveau, *e.g.*, les actions ou les décompositions affecte les connaissances de façon locale et n'est pas répercutée sur les connaissances fondamentales du domaine. Elle a pour but de tester rapidement un nouveau comportement en empêchant le système d'utiliser une action ou une décomposition donnée. Donc, si l'utilisateur désire obtenir un plan sans une action précise ou en évitant d'utiliser un objet donné, il peut supprimer l'action concernée sans pour autant toucher aux connaissances fondamentales du domaine.

L'instruction de modification a été présentée dans le tableau sous forme d'une seule commande pouvant modifier toute la structure, mais en pratique, la modification peut toucher différentes parties des structures et avoir plusieurs applications :

— **Ajout / Relaxation de pré-conditions** : Permet de rajouter ou de supprimer des restrictions au déclenchement des actions et des décompositions. Elle peut être réalisée au niveau des actions ou des décompositions du problème codé pour un test rapide du changement ou être appliquée de façon durable en modifiant les opérateurs et les méthodes.
— **Ajout / Suppression d'effet** : Permet de rajouter ou de supprimer des faits produits ou supprimés par les actions ou les opérateurs. Elle peut être réalisée au niveau des actions pour un test rapide du changement ou être appliquée de façon durable en modifiant les opérateurs.
— **Ajout / Suppression de sous-tâches** : Permet de rajouter ou de supprimer des sous-tâches dans les méthodes et les décompositions. Elle peut être réalisée au niveau de la décomposition pour un test rapide du changement ou être appliquée de façon durable en modifiant la méthode.
— **Modification de la signature** : Permet de modifier le nom ou les paramètres de la structure. Lorsque la modification des paramètres est réalisée sur une méthode ou un opérateur, cela concerne les types des objets affectés par la structure. Si la modification est réalisée sur une décomposition ou une action, cela concerne directement les objets manipulés par la structure.

Lors de l'ajout d'un effet positif ou négatif durant la modification ou l'ajout d'une action, si ce fait fait partie des inerties du domaine, cela entraîne automatiquement une phase d'instanciation et de simplification parce que cela peut conduire à reconsidérer des actions et des méthodes qui ont étés supprimées lors de la phase de simplification.

On peut noter qu'aucune instruction de modification de tâche n'est proposée, cela est dû au fait que ces dernières sont ajoutées, supprimées ou modifiées automatiquement lors de la modification des opérateurs, méthodes, actions et décompositions. Le changement des signatures des décompositions entraîne un traitement spécial, puisque plusieurs de ces deux structures peuvent correspondre à la même tâche composée, et de ce fait en modifier une partie ne va pas modifier la tâche correspondante mais plutôt en créer une nouvelle. Il n'y a pas non plus d'instruction de modification des faits puisque cela affecte forcément toutes les structures du domaine et du problème, il serait donc plus intéressant d'effectuer tous les changements au niveau des connaissances de haut-niveau et de propager tous les changements lors de l'instanciation.

La figure 5.10 montre un exemple d'interaction entre l'utilisateur et le système impliquant des commandes de getting. L'exemple reprend le domaine *rover* avec le problème *p01* utilisé précédemment, avec une petite erreur introduite au niveau du domaine où le prédicat *(empty ?s - STORE)* n'est pas définit dans les effets positifs de l'opérateur *(drop ?x - ROVER ?s - STORE)*. On voit que l'utilisateur définit deux nouvelles tâches à planifier et que le système ne parvient pas à trouver de solution. Il propose alors une piste pour expliquer l'échec grâce à la fonctionnalité du même nom qui sera

abordée en détail dans la section 5.4.6. De l'interaction 3 à 6, l'utilisateur cherche à définir exactement l'erreur en utilisant les fonctionnalités de listing et de getting, il comprend alors que l'erreur vient de l'absence du fait *(empty rover0store)* des effets positifs de l'action *(drop rover0 rover0store)*. Il demande alors au système d'ajouter l'effet en question à l'action à travers une commande de modification, relance une fois la planification avec les mêmes tâches but, et obtient cette fois-ci un résultat. Après avoir confirmé et corrigé l'erreur sur les données de bas-niveau, l'utilisateur demande d'ajouter le prédicat *(empty ?s - STORE)* à l'opérateur *(drop ?x - ROVER ?s - STORE)*, ce qui induit une nouvelle phase d'instanciation et de simplification et la génération d'un nouveau problème codé.

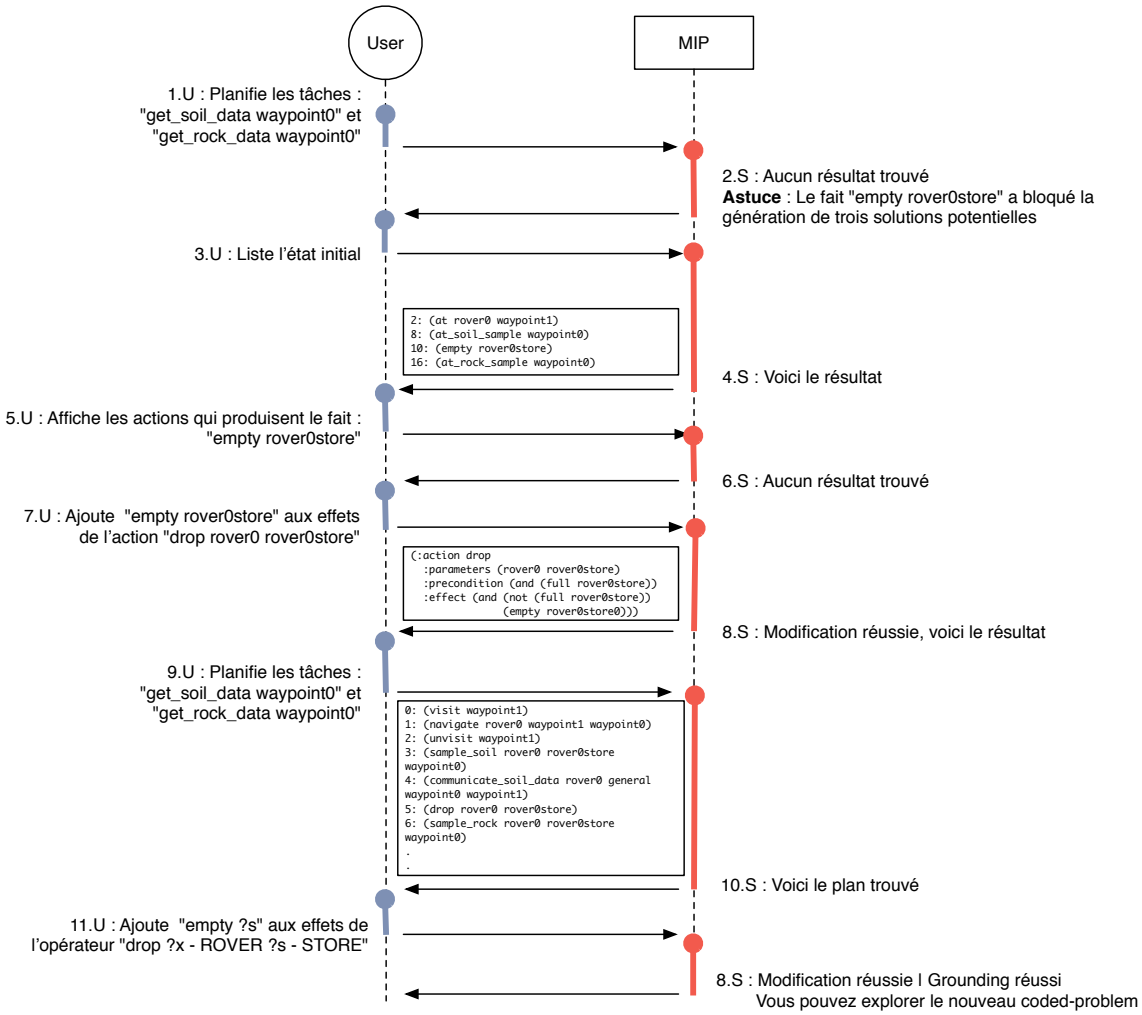


FIGURE 5.10 – Exemple d'échange utilisant les fonctionnalités de modification du domaine.

Cet exemple montre bien que les différentes fonctionnalités de listing, de getting, de co-génération et de modification se combinent très bien afin de mieux guider l'utilisateur dans ces réflexions et ces modifications en lui proposant des explications d'échec, en lui fournissant les données nécessaires à

sa compréhensions, et en lui permettant de réaliser des modifications ciblées dans les connaissances du domaine.

Il est à noter que les structures servants à accélérer les calculs et produites lors de l'instanciation comme les tables des inerties et tableaux d'atteignabilité ne peuvent pas être modifiées directement pour des raisons de cohérence des informations. Ces dernières sont mises à jour automatiquement lors du l'instanciation après modification des autres structures.

5.3.2 Modification du problème et raffinement du but

Alors que la fonctionnalité de modification du domaine guide l'utilisateur dans les changements qu'il peut apporter aux connaissances du domaine sur deux niveaux d'abstraction, la fonctionnalité de modification du problème guide l'utilisateur dans la modification de données liées au problème. Cette fonctionnalité offre la possibilité d'effectuer des changements dans les tâches but, l'état initial et l'état final, d'ajouter, de supprimer ou de modifier des objets du problème. Le tableau 5.4 regroupe l'ensemble des instructions de modification du problème possibles.

Commandes	**Objectif**
[Add | Remove | Modify] the task [signature]	[Ajoute |Supprime| Modifie] la tâche but correspondant à [signature]
[Add | Remove] the object [obj] of type [type]	[Ajoute |Supprime] l'objet [obj] du type [type]
[Add | Remove] the fact [fact] from initial state	[Ajoute |Supprime] le fait [fact] de l'état initial
[Add | Remove] the fact [fact] from goal state	[Ajoute |Supprime] le fait [fact] de l'état but

TABLE 5.4 – Ensemble des commandes de modification du problème de planification

Les instructions de modification du problème peuvent être divisées en deux groupes :

— **Instructions induisant une instanciation et une simplification** Dans cette catégorie, on trouve les instructions d'ajout et de suppression de faits dans l'état initial et les instructions de modification des objets du problème qui permettent d'introduire ou de supprimer les objets utilisés dans toutes les instances des structures de la base de connaissances. Dans le domaine *Rover*,

on peut par exemple définir un nouveau objet *rover* ou en supprimer un pour ne plus l'utiliser. On peut aussi modifier l'état initial en changeant l'emplacement initial du *rover*.

— **Instructions n'induisant pas d'instanciation et de simplification** Dans cette catégorie, on trouve les instructions de modification de la tâche but qui permettent de définir de nouvelles tâches but, de supprimer des tâches existantes et de modifier des objets utilisés pour réaliser ces tâches. Dans le problème 01 du domaine *Rover*, on peut par exemple supprimer la tâche d'échantillonnage du sol, ajouter une tâche de déplacement avant celles qui existent, ou changer de *waypoint* à échantillonner en modifiant les objets des tâches. On trouve aussi les instructions d'ajout et de suppression de faits dans l'état but. Ces deux types d'instructions ne nécessitent pas d'effectuer une instanciation parce qu'elles n'affectent pas le résultat de ce dernier.

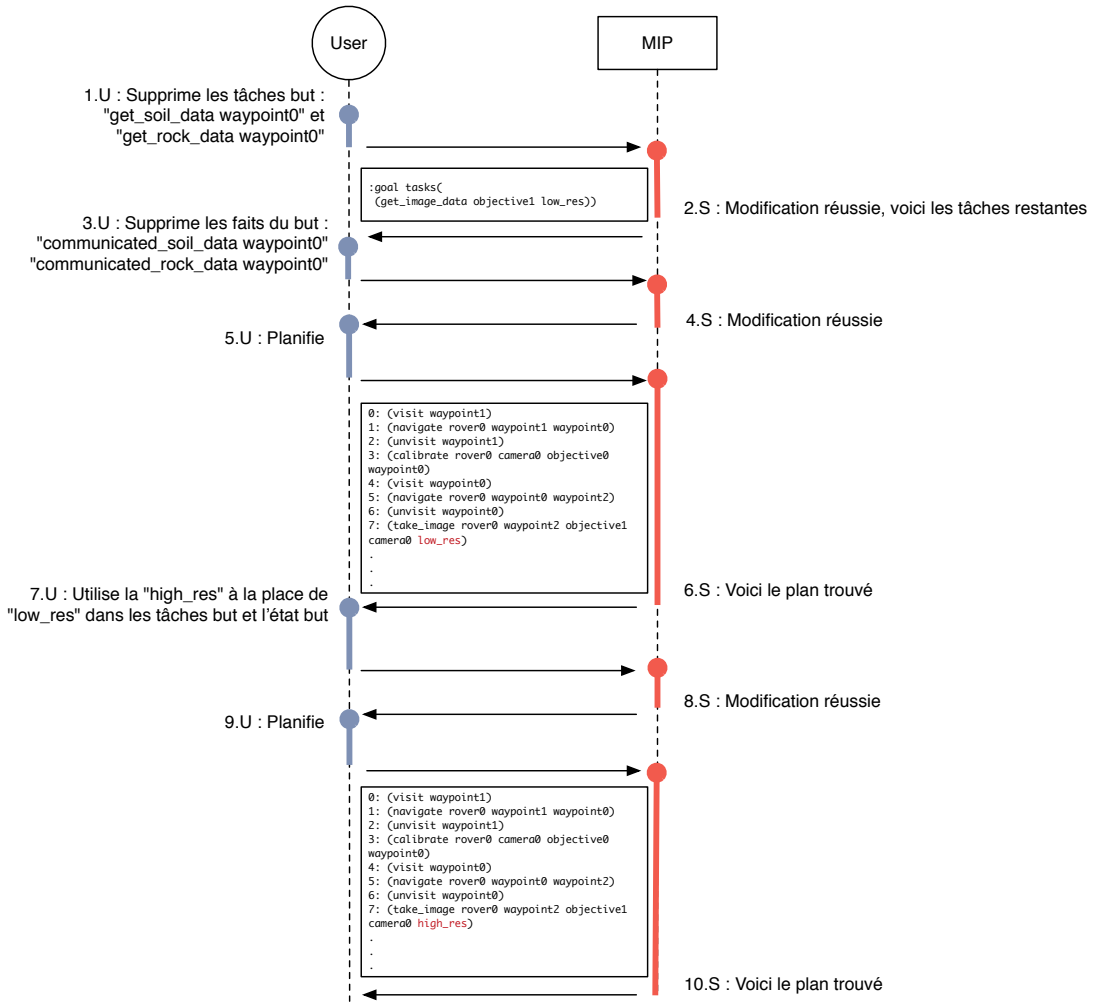


FIGURE 5.11 – Exemple d'échange utilisant les fonctionnalités de modification du problème.

La figure 5.11 représente un exemple d'interaction faisant suite à celui de la figure 5.10 et mon-

trant une utilisation possible des instructions de modification du problème. Dans cet exemple, l'utilisateur modifie les tâches but et l'état but en supprimant des tâches ainsi que des faits de l'état but.Et lance ensuite une planification du problème modifié. Après avoir obtenu un plan résultat, il décide d'utiliser le mode haute résolution pour la prise d'image au lieu du mode basse résolution, et pour cela, il modifie les paramètres de la tâche *"(get_image_data objective1 low_res)"* en remplaçant *"low_res"* par *"high_res"*, modifie le but de la même façon, et lance une autre planification avec le problème modifié.

Les deux modifications de l'exemple n'ont pas nécessité de régénérer un nouveau problème codé, cela n'aurait pas été le cas si l'utilisateur avait rajouté un nouvel objet ou modifié l'état initial. L'utilisateur aurait pu rajouter un deuxième objet *rover*, *e.g. rover2*, avec les prédicats nécessaires dans l'état initial, (*e.g. (at rover2 waypoint1), (available rover2), etc.*). Le système aurait généré une nouvelle version du problème codé en générant toutes les structures utilisant l'objet *rover2*. L'utilisateur aurait eu alors la possibilité d'utiliser le *"rover2* à la place de *"rover0* pour réaliser les mêmes tâches ou de répartir les tâches sur les *rovers*.

5.4 Fonctionnalités de co-construction de plan

5.4.1 Fonctionnalité d'incitation au changement

Lors du processus de recherche de solution, le système MIP possède la capacité d'initier l'interaction avec l'utilisateur afin de lui signaler des solutions incomplètes ou les contraintes qui restent insatisfaites. Et de lui proposer des améliorations qu'il a pu déduire à l'aide des mécanismes d'analyse de solutions potentielles et de méta analyse (section 5.1.4.B et 5.1.4.A. Ces améliorations portent sur la modification :

- Des pré-conditions des actions ou des décompositions afin de lever des contraintes qui bloquent leurs application.
- Des effets des actions afin de rajouter des effets qui manquaient dans la définition initiale et dont l'absence bloquait la génération de solution.
- De l'état but afin de supprimer un fait non atteignable ou ajouter un fait souhaité.
- Des tâches but afin de lever l'obligation de leur réalisation dans le plan et de réduire ainsi la complexité du problème.
- Des sous-tâches des décompositions afin de lever l'obligation de leur réalisation dans le plan et de réduire ainsi la complexité de la planification des tâches liées.

Cette fonctionnalité peut aussi s'avérer très utile pour répondre aux contraintes de *feed-back* im-

médiat et de guidage, surtout lors de planifications avec un grand degré de complexité.

Grâce à l'approche des mécanismes de planification HTN, le planificateur détecte les task-networks primitifs contenant des suites d'actions primitives mais qui possèdent des contraintes non satisfaites conduisant au rejet de cette solution. Lors d'une planification dans un espace d'états, les contraintes non satisfaites peuvent être des propriétés de buts qui ne sont pas produites ou des pré-conditions d'actions qui ne sont pas satisfaites. Alors que dans la planification dans des espaces de plans, les contraintes insatisfaites peuvent être des contraintes before, after, between ou des contraintes d'ordre. Dans les deux cas, ces contraintes sont présentées à l'utilisateur qui peut alors choisir d'accepter ces solutions partielles en levant ces contraintes ou de les refuser et de laisser le système chercher d'autres solutions. Cette fonctionnalité ne peut évidemment être présentée qu'à un utilisateur expert du domaine pour des raisons évidentes de cohérence et de sécurité.

5.4.2 Fonctionnalité de décomposition pas à pas

La fonctionnalité de décomposition pas à pas découle directement du mécanisme d'interaction continue (section 5.1.4.E) et permet à l'utilisateur, dans le cas où il a du mal à comprendre un plan qui lui est proposé ou si il veut suivre une décomposition spécifique tout en analysant les étapes intermédiaires, de décomposer les tâches pas à pas en choisissant la décomposition à appliquer. Cette fonctionnalité est liée a la fonctionnalité d'explication du raisonnement qui récupère toutes les statistiques sur le processus de recherche. A chaque étape, l'utilisateur vérifie l'état courant, les contraintes satisfaites, les pré-conditions et les effets appliqués. Afin de savoir quel décomposition appliquer et comprendre plus facilement les contraintes qui bloquent la génération de plans.

La fonctionnalité de décomposition permet aussi à l'utilisateur de décomposer jusqu'à un certain niveau et de lancer une planification automatique pour trouver toutes les solutions à partir de ce niveau. L'utilisateur peut aussi définir un point d'arrêt sous forme de tâche qui obligera le système à s'y arrêter et de lui demander de choisir la décomposition à appliquer.

5.4.3 Fonctionnalité de réponse immédiate

Cette fonctionnalité permet de répondre à l'aspect bloquant des planificateurs automatiques qui ne fournissent aucun retour avant la fin du processus de recherche du plan solution. À l'aide du mécanisme de planification *any-time* qui sauvegarde le meilleur plan partiel, cette fonctionnalité présente à l'utilisateur la meilleure solution intermédiaire, en guise de réponse à une demande de plan solution durant le processus de recherche et avant la découverte d'une solution. Cette solution peut

alors être analysée par l'utilisateur lui permettant ainsi d'avoir une visibilité sur la direction prise par l'algorithme de recherche et d'effectuer des modifications, s'il les juge pertinentes. Si, par exemple, l'utilisateur demande une solution à un instant donné avant que le planificateur soit capable de trouver une solution, le système lui répond avec la meilleure solution intermédiaire qu'il a exploré jusque là.

La fonctionnalité de réponse immédiate utilise aussi le mécanisme de méta analyse afin de fournir à l'utilisateur des statistiques sur les décompositions et la progression globale du processus de recherche de plan solution. Grâce à la représentation hiérarchique des tâches et avec une exploration ordonnée de l'arbre de recherche, le système est capable de fournir la tâche nécessaire qu'il est entrain de planifier. Toutes ces informations sont offertes à l'utilisateur pour lui permettre de suivre l'état d'avancement du processus de planification.

5.4.4 Fonctionnalité d'amélioration des plans

Elle est directement fondée sur le mécanisme de planification *any-time* qui poursuit, en tâche de fond, la recherche de plans solution après en avoir trouvé. Elle consiste à proposer à l'utilisateur, pour un même problème, des plans qui pourraient être meilleurs que celui ou ceux qui lui ont été proposés auparavant. Le système compare le nouveau plan solution trouvé avec celui qui a déjà été proposé à l'utilisateur, et s'il s'avère être meilleur, en se fondant sur des critères précis, il est proposé comme étant une solution plus performante. Le critère pris en compte, et qui reste le plus facile à évaluer, est la longueur des plans. Plus le plan est court, plus il est performant. Toutes fois, on pourrait facilement imaginer d'autres critères, *i.e.* Des coûts spécifiques pour chaque action, des actions duratives, des décompositions avec préférences, etc. La fonctionnalité d'amélioration continue offre à l'utilisateur la possibilité de choisir le plan qu'il lui convient le plus, en choisissant le plus performant, ou celui qu'il comprend le mieux. Cela offre une meilleure flexibilité dans les solutions et permet de mieux s'adapter aux besoins de l'utilisateur.

5.4.5 Explication du raisonnement

Avec cette fonctionnalité, l'utilisateur a la possibilité de suivre étape par étape le cheminement du processus de recherche de plan solution, depuis le *task-network* initial jusqu'au *task-network* solution. Le planificateur peut consigner dans un registre spécial tous les choix et toutes les actions qu'il a réalisé lors de cette recherche. L'utilisateur pourra alors au choix accéder à ce registre ou demander ces détails lors d'une décomposition pas à pas. Comme ces traces sont très techniques et peuvent

être extrêmement longues, l'utilisateur peut choisir le niveaux de détails dont il a besoin. Le texte ci-dessous montre un exemple de traces consignées par le planificateur lors de la planification de la tâche *do_navigate rover0 waypoint0* du problème 1 du domaine *Rover*. On peut y voire :

— La tâche sélectionnée pour la décomposition sous le titre *«CURRENT TASK»*
— Les décompositions choisies sous le titre *«RELEVENT METHOD»*
— L'acceptation ou le rejet des méthodes avec les mots clés *«ACCEPTED» et «REJECTED»*
— Le résultat de la réécriture du *task-network* sous le titre *«TASK-NETWORK NODE*

Cette fonctionnalité peut s'avérer très utile pour un utilisateur expert afin de comprendre le cheminement au plan solution trouvé, l'impossibilité d'en trouver un ou encore le rejet d'une solution attendue.

5.4.6 Fonctionnalité d'explication de l'échec

Cette fonctionnalité est gérée par un module de gestion des échecs qui permet de collecter les données nécessaires lors de l'étape de planification, de les traiter, et de proposer à l'utilisateur des pistes de réflexion pour comprendre les causes de cet échec et pouvoir y remédier. Considérant l'importance du critère d'explication pour la compréhension et l'acceptabilité de l'outil par l'utilisateur, l'explication présentée à l'utilisateur doit être correctement formalisée et être compréhensible. Pour ce faire, il faut avant tout expliquer qu'est ce qu'un échec dans la planification et les causes possibles.

5.4.6.A Définition de l'échec

Un échec de planification arrive lorsque le planificateur est dans l'incapacité de trouver un plan solution pour le problème qui lui a été fourni après avoir exploré toutes les décompositions pertinentes ou après un délai définit par l'utilisateur.

Le planificateur est incapable de trouver une solution après exploration de tout l'espace de recherche si le domaine a été mal définit. Cela se produit quand :

— Un des opérateurs comporte des erreurs comme :
  — Les pré-conditions de l'opérateur comportent des propositions en trop ;
  — Les effets de l'opérateur ne comportent pas tous les effets nécessaires à l'exécution d'un plan ;
— Une des méthodes de décomposition comporte des erreurs comme :
  — La décomposition est erronée et une tâche a été oubliée ;
  — L'ordre des tâches de la décomposition est incorrecte ;

— La méthode comporte des contraintes qui ne peuvent pas être satisfaites.

Le planificateur peut aussi être dans l'incapacité de trouver une solution si le problème est mal définit, avec un task network initial erroné et/ou un état initial mal renseigné. Dans le cas où le planificateur ne trouve pas de solution dans les délais définis, cela peut être dû aux raisons citées auparavant ou à un problème trop complexe pour le laps de temps imparti.

5.4.6.B Réponse à l'échec

La réponse attendu de la part de l'utilisateur en cas d'échec lors de génération de plan est de premièrement, essayer de comprendre les raisons de l'échec et de procéder ensuite par élimination pour remédier aux raisons qui ont conduit à ce résultat. En cas d'échec après exploration totale, l'utilisateur devra suivre les indications du système pour trouver les raisons qui y ont mené. Mais en cas d'échec dans un délais imparti, l'utilisateur aura potentiellement envie d'étendre le délais d'attente pour voir si le système est capable quand même de sortir un résultat ou, au contraire, il voudra explorer les causes qui ont conduit à ce résultat.

Dans un système en mixed-initiative où l'acceptabilité et la compréhension de l'utilisateur sont des critères fondamentaux, répondre aux différentes réactions de l'utilisateur est crucial. Par conséquent, le module d'explication de l'échec doit pouvoir présenter dans une syntaxe compréhensible par l'utilisateur les causes potentielles de l'incapacité à générer une solution à un instant $t \in]0, t_{imparti}] \cap]0, t_{fin}]$ où $t_{imparti}$ représente la limite du temps imparti et t_{fin} le temps durant lequel s'est terminée la recherche.

5.4.6.C Formalisation de l'explication de l'échec

L'explication d'un échec se présente sous forme d'une probabilité qu'une contrainte soit la cause de cet échec. Elle peut se présenter sous plusieurs formes avec des niveaux d'information plus ou moins détaillés :

— **Explications des échecs résultants des tâches primitives :** Soit le fait $f \in precond(a_1) \cap precond(a_2) \cap ... \cap precond(a_n) \cap$. Où f est est une pré-condition non satisfaite lors de la planification et $a_1, a_2, ..., a_n$ des actions exécutant les tâches $t_1, t_2, ..., t_n$ dont le résultat de chacune se solde par un échec causant le rejet d'un task network différent. l'explication de l'échec

$$exp(e) = \frac{e(f)}{e}$$

où e est l'échec total causé par tous les facteurs et $e(f)$ est l'échec causé par le fact f.

— **Explications des échecs résultants des tâches composées :** Soit la tache t décomposable avec les méthodes $m_1, m_2, ..., m_n$. On dit que t n'a pas été décomposée si

$$\forall m \in \{m_1, m_2, ..., m_n\} : \exists\, constraint(m) = false$$

Dans ce cas l'explication de l'échec peut être présentée comme la tâche t qui n'a pas pu être décomposée avec la liste des contraintes qui n'ont pas pu être satisfaites. Tous les faits f appartenant aux contraintes non satisfaites sont considérés comme ayant causé un échec et sont pris en compte lors du calcul de $exp(e)$.

— **Explications des échecs dans des tasks networks primitifs :** Soit TN un task network primitif qui ne contient que des tâches primitives c-à-d qui représente une solution si toutes ses contraintes sont satisfaites.

$$\forall\, constraint(TN),\ Si\ constraint(TN) = false\ alors\ TN\ est\ rejeté$$

L'échec $e(constraint(tn))$ doit être signalé à l'utilisateur pour lui proposer une solution incomplète qui nécessite de lever la contrainte $constraint(TN)$ pour être acceptée, ensuite,il est ajouté à l'ensemble des échecs qui servira au calcul de $exp(e)$. Il faut noter qu'un échec peut à la fois résulter de tâches primitives et être la cause d'un échec dans un task network primitif.

5.5 Évaluation du système MIP

Nous abordons, dans cette section, une analyse du système MIP en fonction des critères d'évaluation des systèmes de planification en mixed-initiative. Le but à travers cette démarche n'est pas de fournir une évaluation de l'utilisation du système par des utilisateurs comme outil, puisque cela impliquerait une évaluation des outils d'interaction, comme l'interface graphique, et leurs acceptabilité par les utilisateurs. À la place, nous nous concentrons sur les fonctionnalités qu'implémente le système ainsi que les caractéristiques techniques qui permettent de répondre ou non à chacun de ces critères.

5.5.1 Évaluation du guidage

Le système MIP offre plusieurs fonctionnalités pour répondre aux critères de guidage des utilisateurs lors de la co-construction de plans, elles sont listées dans le tableau 5.5. Ainsi, la fonctionnalité d'incitation au changement couplée avec des commandes ciblées pour la modification, l'ajout et la

suppression des données du problème et du but est un outil très puissant pour inciter l'utilisateur à effectuer des modifications dans le problème et permettre ainsi de bien le guider pour éviter des situations d'échec dans le processus de planification.

Le critère de transparence de traitement cherche à évaluer la capacité du système à expliquer toutes les opérations liés à la planification. Afin de répondre à cela, *MIP* est doté d'une fonctionnalité d'explication du raisonnement qui fournit à l'utilisateur un registre rapportant, avec plusieurs niveaux de détails, toutes les actions et les décision qui ont été prises lors du processus de planification. L'utilisateur a ainsi la possibilité d'accéder à ce registre à la fin du processus de planification, ou demander l'affichage de ces informations durant la planification grâce au mécanisme d'interaction. La lecture de ces informations au cours du processus de recherche est facilitée par la représentation des données sous forme hiérarchique et totalement instanciée. Elle permet à l'utilisateur de suivre plus facilement le processus de recherche par rapport à une recherche classique dans un espace d'états.

Critères de guidage	**Fonctionnalités ou caractéristique du système permettant d'y répondre**
Incitation	— Fonctionnalité d'incitation au changement — Fonctionnalités de modification des données, du problème et du but
Transparence de traitement	— Fonctionnalité d'explication du raisonnement — Fonctionnalité de décomposition pas à pas — Le paradigme de décomposition de tâches dans HTN
Groupement	— Fonctionnalité de *getting* des structures du problème — L'aspect hiérarchique de la planification HTN
Feedback immédiat	— Fonctionnalité de réponse immédiate *any-time* — Fonctionnalité d'explication du raisonnement

TABLE 5.5 – Fonctionnalités du système *MIP* répondant aux critères de guidage

À travers les commandes de *getting, MIP* offre un outil très puissant de regroupent des données du problème et du domaine selon un ensemble très varié de critères. Il permet aussi d'exploiter l'aspect

hiérarchique des connaissances définies dans le planificateur HTN puisqu'elles sont déjà regroupées de façon hiérarchique, grâce au lien entre les tâches, les actions, et les décompositions. Ainsi, l'utilisateur peut, par exemple, regrouper les sous-tâches atteignables depuis une tâche donnée. Il peut aussi, regrouper les actions en fonction de leurs effets ou pré-conditions, la même chose pour les pré-conditions des décompositions, etc. Cette capacité de regroupement apporte un bénéfice certain à la compréhension du problème de planification ainsi que les connaissances du domaine.

L'objectif principal du *feedback* immédiat est de montrer à l'utilisateur qu'il y a une évolution dans le processus de planification et que l'interaction avec le système, durant ce dernier, est toujours possible. Dans cette perspective, *MIP* permet de répondre aux périodes d'attentes, durant le processus de recherche de plans, en implémentant la fonctionnalité de réponse immédiate. Cette dernière répond aux questions de l'utilisateur sur l'état d'avancement de la planification avec des statistiques de décomposition de tâches ainsi que les potentielles meilleures solutions trouvées jusqu'à présent. Dans l'absence de chiffres fiables sur le pourcentage de progression de la relecture de planification, la fonctionnalité de réponse immédiate fournit à l'utilisateur un bon moyen pour suivre son évolution.

5.5.2 Évaluation du contrôle et de la charge de travail

Les critères de charge de travail et de décomposition multi-niveaux cherchent à évaluer la charge perceptive de l'utilisateur en ce qui concerne les tâches de planification, ainsi que les tâches de compréhension et de modification des données. Le système *MIP* permet de moduler la charge perceptive de l'utilisateur grâce à la représentation des données avec deux niveaux de détails où l'utilisateur peut accéder à une représentation haut-niveau d'un opérateur définit à l'aide de variable, et accéder aussi aux actions bas-niveau correspondantes à cet opérateur. Les fonctionnalités de *linsing* et de *getting* des données représentent aussi des outils de décomposition de la charge de lecture du problème. Ainsi, si l'utilisateur cherche la définition d'une décomposition spécifique, au lieux de la chercher dans l'ensemble des données, il a la possibilité d'afficher directement cette décomposition, à l'aide de sa signature, ou d'accéder directement à la méthode qui la définit.

Le système *MIP* fournit une fonctionnalité de décomposition de la tâche de planification, grâce au paradigme *HTN* qui permet de décomposer une tâche but en plusieurs sous-tâches dont l'utilisateur peut demander la planification séparément. Par exemple, si le problème de planification consiste à planifier une tâche composée T, et que le plan résultant de cette planification ne convient pas à l'utilisateur ou s'il n'arrive pas à le comprendre, ce dernier peut accéder à une décomposition de cette tâche et demander la planification d'une ou de plusieurs de ces sous-tâches afin de diviser le plan initial et de comprendre la façon dont est planifiée chacune d'entre elles, décomposant ainsi la

charge de compréhension du plan initial.

Proposant actuellement une interface humain/machine simpliste, sous forme de ligne de commande, le système *MIP* ne fournit pas d'outils spécifiques à l'explicitation du contrôle. Ce dernier est toujours en possession de l'utilisateur qui peut lancer une planification, interrompre le processus, interroger le système sur une planification en cours, modifier les données, retourner à une version antérieur des données, etc. Le système peut prendre l'initiative de demander des informations ou de proposer des modifications des données en s'exprimant sous forme textuelle simple sans prendre le contrôle de l'interaction, puisque l'utilisateur peut ignorer tout simplement ces propositions.

L'utilisateur peut ajuster son niveau de contrôle sur le processus de planification. Il peut l'augmenter en utilisant la fonctionnalité de décomposition pas à pas et en intervenant dans le choix de décomposition de toutes les tâches. ou il peut intervenir seulement dans la décomposition d'une tâche spécifique. Ainsi, dès qu'une tâche est choisie pour être décomposée, l'utilisateur est invité à choisir une des décompositions qui lui sont proposées afin qu'elle soit appliquée sur cette tâche. Le niveau le plus faible du contrôle de la planification consiste à automatiser complètement le processus, en laissant le planificateur trouver seul le plan solution. Cette possibilité d'agir sur le niveau d'automatisation du processus de planification participe grandement dans la compréhension des solutions et l'acceptabilité du système par l'utilisateur.

Critères de contrôle et de charge	**Fonctionnalités ou caractéristique du système permettant d'y répondre**
Décomposition du problème	— Le paradigme de décomposition de tâches dans HTN — Représentation multi-niveau des données. — Le *linsing* et le *getting* des données.
Contrôle explicite	- /
Contrôle utilisateur	-/
Automatisation ajustable	— Fonctionnalité de décomposition pas à pas

TABLE 5.6 – Fonctionnalités du système *MIP* répondant aux critères de contrôle et de charge de travail

5.5.3 Évaluation de l'adaptabilité

Grâce à la fonctionnalité d'amélioration des plans, le système *MIP* permet de s'adapter aux besoins et aux préférences de l'utilisateur, en termes de planification. Il permet de proposer à l'utilisateur un plan solution et de poursuivre la recherche d'autres plans qui pourraient être plus intéressants pour l'utilisateur. Il aura ainsi la possibilité de choisir celui qui lui convient le mieux parmi un ensemble de plans. Avec un planificateur HTN produisant des plans totalement ordonnés, *MIP* n'a pas la capacité de fournir des plans partiellement ordonnés offrants à l'utilisateur une plus grande liberté en changeant l'ordre d'exécution des actions ou en les exécutant en parallèle.

Dans le système *MIP*, la fonctionnalité de modification des problèmes, en plus de la division du but sous forme de tâches à décomposer, permet à l'utilisateur d'introduire des exigences dans le *task-network* initial en choisissant un objet spécifique pour une tâche donnée, ou en spécifiant une suite d'actions dans les tâches but afin de les retrouver dans le plan solution. *MIP* s'adapte, ainsi, aux besoins de l'utilisateur qui voudrait obtenir un plan avec certaines exigences. Sans cette capacité, l'utilisateur n'aurait pas la possibilité de spécifier de telles exigences, et serait contraint à explorer toutes les solutions proposées par le planificateur (en admettant que ce dernier le permette).

Critères d'adaptabilité	**Fonctionnalités ou caractéristique du système permettant d'y répondre**
Flexibilité	— La fonctionnalité d'amélioration des solutions — La fonctionnalité de modification du problème et de raffinement du but — Le paradigme de décomposition des tâches dans HTN
Expérience de l'utilisateur	— Fonctionnalité de lecture des données (*listing et getting*) — Fonctionnalité de modification des données — Représentation multi-niveaux des données.

TABLE 5.7 – Fonctionnalités du système *MIP* répondant aux critères d'adaptabilité

MIP offre aussi la possibilité de s'adapter au niveau d'expertise de l'utilisateur à l'aide des fonctionnalités de lecture et de modification des données qui permettent un accès et une manipulation plus ciblée des données. Un utilisateur débutant en planification automatique n'aura pas à accéder

au fichiers *HPDDL* de définition du domaine et du problème pour lire les informations ou les modifier, il aura la possibilité de demander au système de lui afficher les données qu'il désire à l'aide de commande spécifiques. En plus, le système lui permet de lire et de modifier les données de bas niveaux sous forme d'actions et de décompositions (instanciées) et de voir des liens qu'il n'a pas la capacité de déduire.

5.5.4 Évaluation de la gestion des erreurs

Le système *MIP* prend en charge la gestion de plusieurs types d'erreurs qui peuvent être introduites dans les données du domaine et du problème. Cela est réalisé à travers l'analyse syntaxique et sémantique automatique des données en entrée, à partir des fichiers *HPDDL* ou des saisies de l'utilisateur. Cela passe aussi par une analyse plus approfondie des liens entres les données, rendue possibles grâce au calculs d'atteignabilité et des inerties.

Afin de prévenir les erreurs de saisie, de définition de doublons, d'introduction de conflits dans les pré-conditions, les effets, les contraintes, *etc.*, le système *MIP* réalise des analyses syntaxiques et lexicales sur les données d'initialisation lors de l'étape de chargement des données, en amont de l'étape de *grounding*. Cette vérification est aussi réalisée tout au long des interactions de modification, à travers le gestionnaire de modification des données. Si la modification est de haut niveau et nécessite une nouvelle étape de *grounding*, l'analyse est réalisée, de la même manière que lors du chargement des données d'initialisation, sur les données définies en *PDDL*. Si la modification est de bas niveau et n'affecte pas toutes les données, l'analyse est réalisée sur la partie des données qui a été modifiée. Par exemple, si l'utilisateur modifie un opérateur en ajoutant un prédicat dans ses effets, l'analyse de toutes les données est réalisée, mais si l'utilisateur rajoute un fait aux effets d'une action, seule cette dernière va être analysée.

L'analyse syntaxique et sémantique des données ne suffit pas à avertir l'utilisateur de toutes les erreurs ou de la portée des effets de ses modifications. Par exemple, s'il ajoute un fait aux effets d'une action et que ce dernier appartient à la table des inerties calculées avant modification, les données du problème instancié sont incohérentes et introduisent des conflits. Le système avertit l'utilisateur que la modification réalisée affecte les inerties du problème et recalcule un nouveau problème instancié ou annule la modification, en fonction de sa réponse. Lorsqu'une erreur est détectée suite à une modification des données, le système peut :

— Signaler l'erreur et bloquer directement la modification si l'erreur est syntaxique, sémantique, ou si elle produit une incohérence des données qui peut être détectée sans *grounding*. Cela se produit par exemple si l'utilisateur utilise un objet ou un fait qui n'est pas déclarés, ou s'il

ajoute un fait, contenu dans la table des inerties aux effets d'une action.

— Laisser l'utilisateur la corriger en affichant textuellement l'erreur, son type son emplacement dans la définition du domaine ou du problème. Cela se produit lorsque l'utilisateur modifie des données de haut-niveau et que ces dernières ont été reportées par le système sur les fichiers PDDL.

— Proposer à l'utilisateur de revenir à la version des données précédant la modification.

Les fonctionnalités citées précédemment permettent de palier aux erreurs liées à la sémantique et à la cohérence des données d'entrée. Un autre type d'erreur est possible. Ce sont les erreurs liées à la définition des données des connaissances et l'expression du problème. Ce genre d'erreur est très difficile à déceler et nécessite d'effectuer une planification (qui peut être très longue) et qui aboutirait à un échec de génération de plan solution. Par exemple, si l'utilisateur demande au système de planifier une tâche et que le système lui répond qu'aucun plan solution n'a été trouvé, l'utilisateur doit effectuer un déboggage extrêmement compliqué afin de trouver la source du problème. Afin de palier à cette situation, *MIP* propose, en plus des fonctionnalités de décomposition pas à pas et du registre d'exécution, une fonctionnalité d'explication de l'échec qui calcul des statistiques sur les solutions potentielles rejetées et propose un ensemble de causes possibles, afin de guider l'utilisateur pour corriger cette situation.

Critères de gestion des erreurs	**Fonctionnalités ou caractéristique du système permettant d'y répondre**
Protection contre les erreurs	— Fonctionnalité de modification des données — Analyse syntaxique et sémantique des données d'initialisation.
Correction des erreurs	— Fonctionnalité d'explication de l'échec — Versionning des *coded-problems*.
Qualité des messages	— Affichage textuel des erreurs avec affichage des données concernées et leurs définitions dans les fichiers PDDL. — Analyse des commandes par le gestionnaire d'interaction et signalement des erreurs avec affichage des commandes possibles .

TABLE 5.8 – Fonctionnalités du système *MIP* répondant aux critères de gestion des erreurs

La combinaison des fonctionnalités de modification des données avec l'analyse syntaxique et sémantique, ainsi que l'anticipation des changements offre un ensemble d'outils très puissants afin de palier à l'introduction d'erreurs dans les données du domaine et la définition du problème de planification. De plus, l'accès ciblé à l'aide des fonctionnalités de modification des données, l'explication de l'échec et le versionning des problèmes permettent de faciliter la correction de toutes les erreurs restantes.

5.5.5 Évaluation de la cohérence

Grâce à l'exploitation de la représentation hiérarchique des connaissances qui guide la recherche de plan solution, à travers la décomposition successive des tâches, des changements mineurs dans les données conduisent à des changement mineurs dans les solutions. Par exemple, Si le problème de planification contient quatre tâches but (T_1, T_2, T_3 et T_4) à planifier et que l'utilisateur remplace la tâche T_3 par une nouvelle tâches $T_3^{'}$. Cela a de fortes chances que seule la partie du plan qui résulte de la décomposition de cette tâche soit affectée, contrairement à une technique de planification classique guidée par une heuristique de recherche qui peut trouver un plan complètement différent en changeant un fait de l'état but. Cela s'explique par le fait que cette dernière explore un espace d'états plus vaste, en étant guidée par une fonction heuristique qui a une forte dépendance du but. Cependant, *MIP* ne prévoit pas de mécanismes spécifiques

MIP permet de répondre au critère de signifiance des dénominations grâce à l'analyse syntaxique et sémantique réalisée lors du chargement des données, des modifications de haut-niveau, nécessitant un nouvelle instanciation et simplification, et lors des modifications de bas-niveau. Lors de ces opération, le système s'assure, à travers l'interpréteur du problème et le gestionnaire de modification, de l'unicité des noms des objets, des types, des prédicats, des tâches, ainsi que des opérateurs et des actions. Rendant les dénominations facilement lisibles par l'utilisateur. Les méthodes et les décompositions peuvent être dédoublées représentant chacune une façon différente de réaliser une tâche. Le système permet aussi d'avoir une lisibilité plus facile des opérateurs et des méthodes à travers la considération de l'unicité des variables, mise en place dans *PDDL 3.0* et qui n'existe pas dans le langage des planificateurs *SHOP*. Elle consiste à considérer deux variables portant le même nom à l'intérieur d'un même opérateur ou méthode comme étant identiques, *i.e.* lors de l'instanciation et la simplification, elles ne peuvent pas avoir deux valeurs différentes. Et à l'inverse, si elle ont des noms différents, elle doivent avoir des valeurs différentes.

Critères de cohérence	**Fonctionnalités ou caractéristique du système permettant d'y répondre**
Stabilité	— Le paradigme de décomposition des tâches dans HTN
Comparaison	— Fonctionnalité de comparaison de plans
Signifiance des dénominations	— Le typpage des variables et l'unicité des objets — L'inférence des types de l'algorithme *GTOHP*

TABLE 5.9 – Fonctionnalités du système *MIP* répondant aux critères de cohérence

5.5.6 Évaluation technique

Le tableaux 5.10 présente un résumé des caractéristiques techniques du système *MIP* qui ont été abordées tout au long de la deuxième partie de ce manuscrit.

Afin de résumer les points abordés précédemment, la figure 5.12 présente, sous forme d'un diagramme en toile d'araignée, le degré de satisfaction du système *MIP* aux critères d'évaluation des systèmes en mixed-initiative. Chaque axe du diagramme représente le degré de satisfaction d'un des critères d'évaluation [10] et se décompose en quarte niveaux : non satisfait, faiblement satisfait, moyennement satisfait et fortement satisfait.

Globalement, le système *MIP* satisfait fortement aux critères d'évaluation, avec une grande lacune au niveau du contrôle explicite qui n'est pas bien géré, à cause notamment de l'interface d'interaction en ligne de commandes. La gestion d'autres critères peut aussi être améliorée :

— Pour plus de flexibilité, le système devrait proposer des plans partiellement ordonnés afin d'augmenter la flexibilité d'exécution et d'ordonnancement des actions.
— Pour plus de stabilité dans les solutions proposée, on pourrait imaginer un mécanisme de planification spécifique, qui s'approcherait le plus possible de la décomposition des tâches suivie lors de la planification du plan précédent, en détectant les parties du plans précédent

10. Tous les critères d'évaluation des système de planification en mixed-initiative sont représentés dans le diagramme, à l'exception des critères techniques qui ne peuvent pas être évalués en degré de satisfaction, mais qui servent à présenter les caractéristiques techniques des systèmes

caractéristique technique	**Fonctionnalités ou caractéristique du système permettant d'y répondre**
Technique de planification	— Planification HTN sur des données instanciées
Interface de communication	— Interface textuelle en ligne de commandes
Représentation des solutions	— Présentation textuelle des plans sous forme de liste
Édition des connaissances	— Modification ciblée sur deux niveaux d'abstraction — Fonctionnalité de modification des données du domaine et du problème

TABLE 5.10 – Résumé des caractéristiques techniques du système *MIP*

qui peuvent réaliser une partie des tâches but, et planifiant les tâches qui restent, tout en faisant le lien entre les parties du plan final et en s'assurant de sa validité. Cependant, la mise en place d'un tel mécanisme est une tâche très complexe qui nécessiterait beaucoup de recherche théorique et d'implémentation

— Pour une meilleure comparaison des plans solutions, d'autres critères, en plus de la longueur du plan pourraient être pris en compte, en rajoutant par exemple un coût comme facteur de pondération des actions, une probabilité d'acceptation du plan en fonction de l'historique de l'utilisateur, etc.

Conclusion

Dans ce chapitre, nous avons présenté le système de planification en mixed-initiative utilisant une approche de planification HTN totalement instanciée et augmentée d'une interaction constante avec l'utilisateur pour la compréhension et la modification des connaissances du domaine et du problème ainsi que des fonctionnalités de co-génération de plans en mixed-initiative. La description du système et de ces fonctionnalités a permis de mettre en avant la valeur ajoutée apportée par le pro-

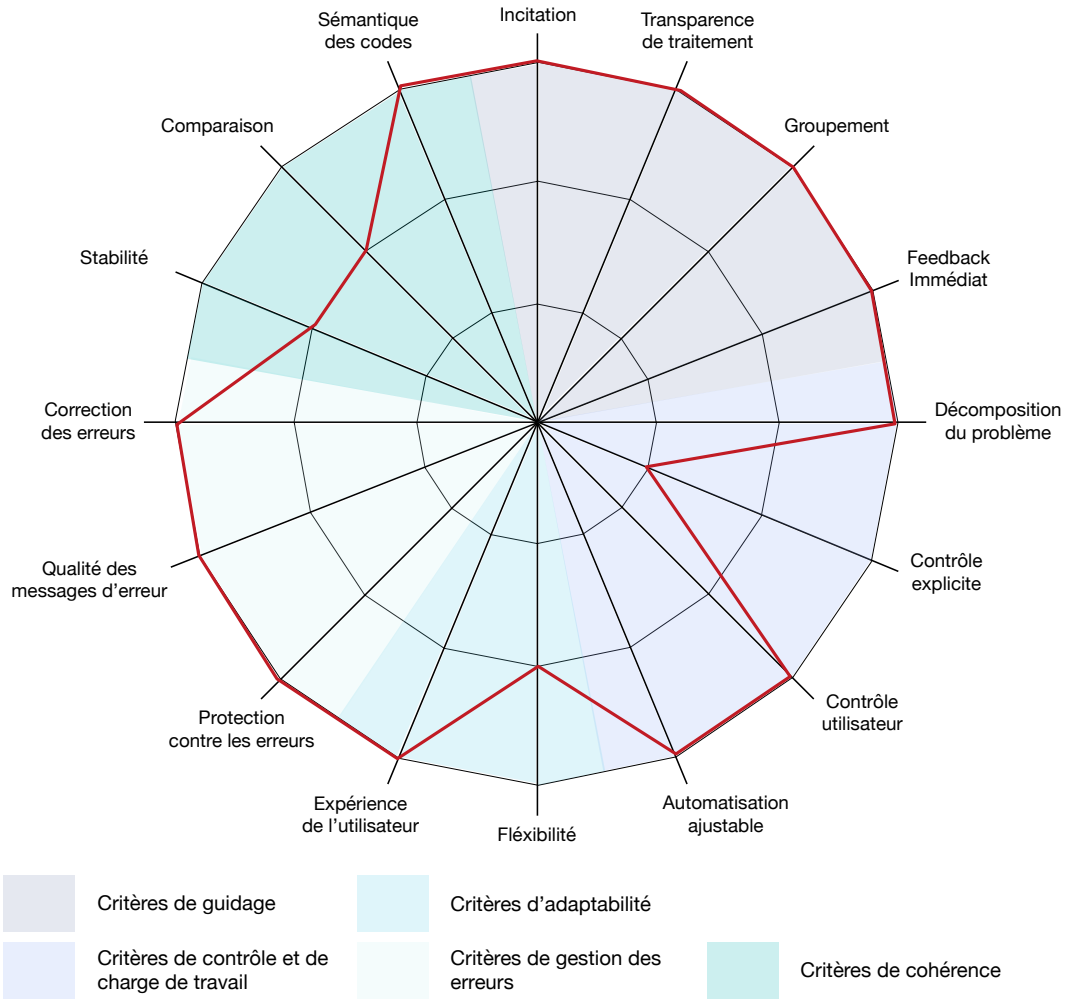


FIGURE 5.12 – Diagramme résumant le degré de satisfaction du système *MIP* des critères d'évaluation des système de planification en mixed-initiative

cessus d'instanciation de simplification et sa pertinence dans un processus qui implique l'utilisateur avec sa capacité d'abstraction dans le mécanisme de planification.

Dans un premier temps, nous avons introduit le système, ensuite, nous avons présenté son architecture globale et les détails de ses différents modules ainsi que leurs interactions. Ensuite, nous avons expliqué les fonctionnalités de co-génération de plan et leurs bénéfices par rapport à une planification classique.

L'organisation de l'architecture du système sous une forme modulaire lui offre une meilleure possibilité d'évolution et un rajout plus facile de nouvelles fonctionnalités. De plus, l'instanciation et la simplification des problèmes de planification avec une représentation des connaissances et une in-

teraction sur deux niveaux d'abstraction s'inscrit très bien dans l'esprit d'adaptation au besoins et au niveau de l'utilisateur souhaitée dans les systèmes en mixed-initiative. Il reste maintenant à savoir si les composants du système, leur organisation ainsi que les fonctionnalités fournies permettent de répondre aux critères exigés pour la construction d'un système de planification en mixed-initiative. Cette question sera débattue dans le chapitre suivant.

6

Conclusion générale

Le développent d'outils d'assistance à la gestion et à la configuration des systèmes complexes requière la prise en compte de plusieurs facteurs : L'outil doit permettre de modéliser les connaissances sur le domaine d'application dans une représentation qui soit compréhensible par l'utilisateur, il doit aussi être assez performant afin de répondre le plus vite possible aux requêtes de l'utilisateur, enfin, les choix de raisonnement de l'outil et les solutions qu'il propose doivent être facilement compréhensibles et acceptables par ses utilisateurs. Dans cette perspective, notre objectif dans cette thèse a été de développer un outil d'assistance à la gestion et à la configuration des systèmes complexes fondé sur une approche de planification automatique en mixed-initiative.

Nous avons soutenu que la planification automatique permettait de répondre au besoin de génération automatique de procédures, à travers la génération d'une suite d'actions applicables depuis un état initial pour atteindre un état but. Cependant, dans son approche classique, la planification automatique n'exploite pas assez ou trop peu les connaissances procédurales que peut apporter un utilisateur expert du domaine afin de guider l'algorithme de recherche. Nous nous somme ainsi orienté vers la planification HTN qui permet d'exploiter des méthodes de décompositions de tâches. Nous avons fait le constat que la planification HTN n'offrait pas de performances suffisantes pour une interaction fluide avec l'utilisateur et qu'on pouvait y apporter une amélioration importante à travers une étape de pré-traitement sur les connaissances du problème. Enfin, l'implication de l'utilisateur dans la recherche de solution est apparu comme un pré-requis à sa compréhension et son acceptation de la solution qui lui est proposée et nous nous somme ainsi orientés vers la planification en mixed-initiative. Toutes ses approches scientifiques et ces choix techniques ont conduit aux contributions suivantes :

1. Nous avons proposé et implémenté un algorithme d'instanciation et de simplification des problèmes *HTN* qui permet de réduire considérablement leur complexité. Comme on a pu le constater, cela a pour effet d'améliorer de façon très significative les performances du planificateur *HTN* en terme de temps de traitement sans affecter de manière négative ses performances en terme de longueur de plan. L'étape d'instanciation, permet d'analyser les données du problème et d'effectuer des calculs d'atteignabilité qui pourraient être exploités dans la perspective de proposer une heuristique de recherche plus performante.
2. Nous avons par la suite développé le planificateur *GTOHP* qui implémente une version de l'algorithme *SHOP* prenant en entrée un problème totalement instancié et simplifié. Nous avons aussi démontré empiriquement, à l'aide de ce planificateur, que le processus d'instanciation et de simplification améliore les temps de traitement des algorithmes de planification *HTN*.
3. Nous avons proposé *HPDDL*, qui est une extension hiérarchique du langage de planification *PDDL* permettant de définir des méthodes de décomposition de tâches et des problèmes HTN

sur des problèmes *PDDL* classiques. Elle offre la possibilité d'étendre directement des problèmes classiques avec des connaissances hiérarchiques et de faire appel soit à des algorithmes de planification prenant en entrées des problèmes *PDDL* ou des algorithmes *HTN*. En plus de cela, le langage *HPDDL* définit plusieurs types de contraintes possibles dans une méthode, ce qui permet d'utiliser des algorithmes de recherche *HTN* dans un espace d'états, ou dans un espace de plans.

4. Nous avons proposé et implémenté le système *MIP* qui est un outil de planification automatique en mixed-initiative indépendant du domaine d'application. Cet outil comporte plusieurs modules et inclut une version du planificateur *GTOHP* dotés de mécanismes de récolte de données et des points d'actions pour guider la recherche de plan solution. Il implémente aussi des modules d'interaction qui guident l'utilisateur dans la lecture et la modification du problème, ainsi que dans la recherche de solution et le déboggage des erreurs. L'interaction au niveau du problème se fait sur deux niveaux d'abstraction rendu possible grâce à l'étape d'instanciation et de simplification. Nous avons aussi proposé une grille d'évaluation des systèmes de planification en mixed-initiative inspirés des critères d'ergonomie. Enfin, nous avons montré, à travers une évaluation point par point que le système *MIP*, avec ses fonctionnalités de lecture, de modification et de planification en mixed-initiative, répondait très bien à ces critères.

Dans l'état d'avancement actuel des travaux, on peut énumérer certaines limites qui sont liées en grande partie à des contraintes de développement et de durée de thèse :

— Du point de vue de la planification automatique, le langage *HPDDL* et l'algorithme d'instanciation et de simplification *HTN* ne prennent pas en compte des notions qui peuvent s'avérer très utiles dans la gestion et la configuration des systèmes complexes, comme : les variables numériques qui servent à représenter des ressources dont la valeur ne peut pas être binaire (e.g. mémoire de stockage disponible, consommation énergétique, distance, poids, etc.), les actions duratives, *i.e.*, des actions dont l'application n'est pas instantanée et qui peuvent être utilisées dans la planification temporelle, et la planification avec préférences qui permettrait de définir des préférences de décomposition et de donner la priorité à certaines actions par rapport à d'autres. Nous nous somme, aussi, peu intéressé à l'exécution réelle des plans produits par l'outil. Actuellement, ce dernier produit un plan solution qui doit être analysé, validé, puis exécuté par l'utilisateur. Donc c'est lui qui assure que l'application du plan est toujours valide dans l'état actuel du système et initie une nouvelle tâche de planification avec les données à jour si ce n'est plus le cas.
— Du point de vue de l'interaction, nous nous somme focalisé sur les mécanismes internes de contrôle des connaissances et de la planification sans nous attarder sur l'interface de commu-

nication entre l'utilisateur et la machine. Ainsi, nous avons développé une interface textuelle avec des commandes simples afin de permettre, d'un coté, à l'utilisateur de définir le but, d'interroger et de modifier les connaissances ; et de l'autre coté, au planificateur de faire remonter des informations pertinentes pour la planification en cours et de proposer des explications et des corrections potentielles. La simplicité de l'interface actuelle représente un frein aux interactions qui peuvent avoir lieu entre le planificateur et l'utilisateur, en rendant notamment très difficile l'affichage hiérarchique des tâches lors de la recherche de plans ou dans les solutions. En plus, une interface en ligne de commande représente un point bloquant pour bon nombre d'utilisateurs non familiers avec ce genre de représentation.

— Du point de vue de l'évaluation, nous avons effectué une série d'expérimentations afin d'évaluer les performances de *GTOHP* et des apports de l'algorithme d'instanciation. Cependant, il ne nous a pas été possible d'évaluer de façon expérimentale le système *MIP*, en grande partie à cause de la difficulté d'évaluation du système et de l'interaction avec des utilisateurs sans que tous les résultats soient influencés de façon négative par l'interface en ligne de commandes.

Un certain nombre d'amélioration peuvent être apportées aux travaux réalisés durant cette thèse, et beaucoup de travaux peuvent être mis en perspectives au vu des résultats obtenus :‘

— Grâce à l'expressivité du langage *HPDDL* et à son intégration dans la librairie de planification *PDDL4J*, il est maintenant possible d'implémenter des algorithmes *HTN* ou non, utilisant en entrée différents types de problèmes, instanciés ou non. Grâce à cette polyvalence, on a la possibilité de comparer différentes approches de planification automatique sur la base d'un même langage de programmation et des mêmes structures de données, ainsi les résultats d'éventuelles expérimentations ne seront pas affectés par ces paramètres. De plus, il serait intéressant d'explorer l'approche de définition incrémentale des connaissances du domaine *HTN*. On peut par exemple implémenter des algorithmes du type *GoDel* qui prennent en entrée des problèmes de planification *PDDL* classiques et sur lesquels on pourrait rajouter de la connaissance sous forme de méthodes de décomposition et choisir d'utiliser comme but soit un état à atteindre ou une tâche à réaliser. Cette approche peut s'avérer être très utile pour un outil d'aide à la décision indépendant du domaine, utilisé dans des applications concrètes dont la formalisation des connaissances de décomposition est définie au cours des interactions avec les experts.
— L'expressivité du langage *HPDDL* peut être aussi grandement améliorée avec la prise en compte des variables numériques, des actions duratives, et de la définition des préférences dans les méthodes du domaine de planification. Définir des variables numériques et des actions est déjà possible dans *PDDL*. Cependant, étendre cela aux méthodes *HTN* résulterait par la dé-

finition de durées maximums et minimums pour la réalisation d'une tâche. cela pourrait se faire selon deux approches : (1) la mise à jour du langage *HPDDL* afin de définir la durée lors de la définition des méthodes ; (2) le calcul automatique des ces durées en fonction de celles définies dans les opérateurs. La définition des préférences dans les méthodes permettrait de donner la priorité à certaines décompositions en fonction de l'état dans lequel elles seraient appliquées. Cette prioritisation des décompositions rajoute encore plus d'expressivité au langage et se rapproche plus du raisonnement de l'utilisateur qui peut par exemple préférer une séquence d'actions plus robuste même si elle est plus longue et plus coûteuse.

— Les résultats de l'évaluation du planificateur *GTOHP* ont montré qu'il est plus performant en terme de temps traitement que d'autres algorithmes HTN équivalent. Cependant, ces performances dépendent beaucoup de la définition des méthodes de décomposition et des contraintes qui y sont liées. On peut alors envisager d'inclure les contraintes *after* dans les méthodes des algorithmes HTN à espace d'états (qui n'utilise normalement que des pré-conditions sous forme de contraintes *before*). Grâce aux calcules d'atteignabilité qui peuvent être réalisés lors de l'étape d'instanciation et de simplification, on peut savoir quels sont les faits qui peuvent être produits ou consommés par les sous tâches de cette décomposition et de vérifier par la suite si les contraintes de type *after* peuvent être satisfaites. On peut aussi investiguer la possibilité de mise en place d'une heuristique de décomposition qui exploite les calculs d'atteignabilité afin de choisir la meilleure décomposition en premier ou élaguer certaines branches de l'arbre de recherche.

— Pour finir, l'interface de communication de *MIP* qui se présente sous forme de lignes de commandes représente un point critique à améliorer à travers la mise en place d'une interface graphique organisant les différents points d'interaction sous forme d'onglets graphiques, ergonomiques et facilement consultables. On peut imaginer avoir un onglet par fonctionnalité du système. L'utilisateur aura, par exemple, un onglet pour éditer le domaine et les problèmes, un onglet pour afficher et interroger le système sur les connaissances du domaine, un onglet pour suivre la recherche de solution, consulter les logs et statistiques récoltées, un onglet pour explorer les solutions et les modifications proposée, etc. Cette interface devra être le sujet d'une étude de conception poussée afin d'optimiser les critères d'évaluation des systèmes en mixed-initiative qui y sont directement liés, comme les critères de guidage, de contrôle et de charge de travail.

Bibliographie

Aha, D W., Breslow, L A., & Muñoz-Avila, H. 2001. Conversational case-based reasoning. *Applied Intelligence*, **14**(1), 9–32.

Ai-Chang, Mitchell, Bresina, John, Charest, Len, Jonsson, Ari, Hsu, Jennifer, Kanefsky, Bob, Maldague, Pierre, Morris, Paul, Rajan, Kanna, & Yglesias, Jeffrey. 2003. *MAPGEN : mixed initiative planning and scheduling for the Mars' 03 MER mission*. Pasadena, CA : Jet Propulsion Laboratory, National Aeronautics and Space Administration.

Ai-Chang, Mitchell, Bresina, John, Charest, Len, Chase, Adam, Hsu, Jennifer Cheng-jung, Jonsson, Ari, Kanefsky, Bob, Morris, Paul, Rajan, Kanna, Yglesias, Jeffrey, *et al.* . 2004. Mapgen : mixed-initiative planning and scheduling for the mars exploration rover mission. *Intelligent Systems, IEEE*, **19**(1), 8–12.

Alford, R., Bercher, P., & Aha, D W. 2015. Tight Bounds for HTN Planning with Task Insertion. *Pages 1502–1508 of : Proceedings of the International Joint Conference on Artificial Intelligence*.

Allen, J., & Ferguson, G. 2002. Human-machine collaborative planning. *Pages 27–29 of : Proceedings of the Third International NASA Workshop on Planning and Scheduling for Space*.

Allen, J., Tenenberg, J., Pelavin, R., & Kautz, H. 1991. *Formal models of reasoning about plans*.

Allen, J. F, & Koomen, J. 1983. Planning Using a Temporal World Model. *Pages 711–714 of : IJCAI*, vol. 8.

Amol, D M., & Subbarao, K. 1998. Encoding HTN Planning in Propositional Logic. *Pages 190–198 of : Proceedings of the International Conference on Artificial Intelligence Planning Systems*.

Barreiro, Javier, Boyce, Matthew, Do, Minh, Frank, Jeremy, Iatauro, Michael, Kichkaylo, Tatiana, Morris, Paul, Ong, James, Remolina, Emilio, Smith, Tristan, *et al.* . 2012. EUROPA : a platform for AI planning, scheduling, constraint programming, and optimization. *4th International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS)*.

Belker, T., Hammel, M., & Hertzberg, J. 2003. Learning to optimize mobile robot navigation based on HTN plans. *Pages 4136–4141 of : Proceedings of the International Conference on Robotics and Automation*, vol. 3. IEEE.

Bevacqua, Giuseppe, Cacace, Jonathan, Finzi, Alberto, & Lippiello, Vincenzo. 2015. Mixed-Initiative Planning and Execution for Multiple Drones in Search and Rescue Missions. *In : Twenty-Fifth International Conference on Automated Planning and Scheduling*.

Blum, A., & Furst, M. 1997. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence*, **90**(1-2), 281–300.

Bresina, John L, & Morris, Paul H. 2007. Mixed-initiative planning in space mission operations. *AI magazine*, **28**(2), 75.

Bresina, John L, Jónsson, Ari K, Morris, Paul H, & Rajan, Kanna. 2005. Mixed-Initiative Activity Planning for Mars Rovers. *Pages 1709–1710 of : IJCAI*. Citeseer.

Carbonell, J R. 1970. AI in CAI : An artificial-intelligence approach to computer-assisted instruction. *IEEE transactions on man-machine systems*, **11**(4), 190–202.

Carbonell, J R., & Collins, A M. 1970. *Mixed-initiative systems for training and decision-aid applications*. Tech. rept. DTIC Document.

Chen, K., Xu, J., & Reiff-Marganiec, S. 2009. Markov-htn planning approach to enhance flexibility of automatic web service composition. *Pages 9–16 of : Proceedings of the International Conference on Web Services*. IEEE.

Chu-Carroll, J., & Brown, M. 1997. Initiative in collaborative interactions-its cues and effects. *In : Proc. of AAAI Spring*.

Cohen, R., Allaby, C., Cumbaa, C., Fitzgerald, M., Ho, K., Hui, B., Latulipe, C., Lu, F., Moussa, N., Pooley, D., *et al.* . 1998. What is initiative ? *User Modeling and User-Adapted Interaction*, **8**(3), 171–214.

Cortellessa, G., & Cesta, A. 2005. Towards a Reliable Evaluation of Mixed-Initiative Systems. *In : Mixed-Initiative Problem-Solving Assistants : Papers from the AAAI Fall Symposium*.

Cox, Michael T, & Veloso, Manuela M. 1997. Supporting combined human and machine planning : An interface for planning by analogical reasoning. *Pages 531–540 of : Case-Based Reasoning Research and Development*. Springer.

Cox, Michael T, & Veloso, Manuela M. 1998. Goal transformations in continuous planning. *Pages 23–30 of : Proceedings of the 1998 AAAI fall symposium on distributed continual planning*.

Cox, Michael T, & Zhang, Chen. 2007. Mixed-initiative goal manipulation. *AI Magazine*, **28**(2), 62.

Currie, K., & Tate, A. 1991. O-Plan : the open planning architecture. *Artifical Intelligence Journal*, **52**(1), 49–86.

Dagkli, E., & Refanidis, I. 2009. MIXPLAN : A CLP-Based Mixed-initiative Planning System for Temporal Domains. *Pages 73–78 of : Fourth Balkan Conference in Informatics*. IEEE.

de la Asunción, M., Castillo, L., Fdez-Olivares, J., García-Pérez, Ó., González, A., & Palao, F. 2005. SIADEX : An interactive knowledge-based planner for decision support in forest fire fighting. *Artificial Intelligence Communications*, **18**(4), 257–268.

Dimopoulos, Y., Gerevini, A., Haslum, P., & Saetti, A. 2006. The benchmark domains of the deterministic part of IPC-5. *Abstract Booklet of the competing planners of ICAPS-06*, 14–19.

Erol, K., Hendler, J. A, & Nau, D S. 1994. UMCP : A Sound and Complete Procedure for Hierarchical Task-network Planning. *Pages 249–254 of : AIPS*, vol. 94.

Ferguson, G., & Allen, J. 2007. Mixed-initiative systems for collaborative problem solving. *AI magazine*, **28**(2), 23.

Ferguson, G., Allen, J F., Miller, B W., *et al.* . 1996. TRAINS-95 : Towards a Mixed-Initiative Planning Assistant. *Pages 70–77 of : AIPS*.

Ferguson, G., Allen, J F., *et al.* . 1998. TRIPS : An integrated intelligent problem-solving assistant. *Pages 567–572 of : AAAI/IAAI*.

Fikes, R E., & Nilsson, N J. 1971. STRIPS : A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, **2**(3-4), 189–208.

Fox, M., & Long, D. 2003. PDDL 2. 1 : An extension to PDDL for expressing temporal planning domains. *Journal of artificial intelligence research*, **20**, 61–124.

Frank, Jeremy, & Jónsson, Ari. 2003. Constraint-based attribute and interval planning. *Constraints*, **8**(4), 339–364.

Georgievski, I., & Aiello, M. 2015. HTN planning : Overview, comparison, and beyond. *Artifical Intelligence Journal*, **222**, 124–156.

Ghallab, M., Nau, D., & Traverso, P. 2004. *Automated planning : theory & practice*. Elsevier.

Giampapa, Joseph A, & Sycara, Katia. 2001. Conversational case-based planning for agent team coordination. *Pages 189–203 of : International Conference on Case-Based Reasoning*. Springer.

Hartanto, R. 2011. Fusing DL Reasoning with HTN Planning. *KI-Künstliche Intelligenz*, **25**(1), 81–84.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, **26**, 191–246.

Immaneni, Trivikram, & Cox, Michael T. 2004. GTrans : An application for managing mixed-initiative collaborative planning during emergency response situations. *Pages 121–126 of : Proceedings of the 2004 International Symposium on Collaborative Technologies and Systems : CTS 2004*.

Jiang, H., Duerstock, B S., & Wachs, J P. 2014. A machine vision-based gestural interface for people with upper extremity physical impairments. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, **44**(5), 630–641.

Jonsson, KA, Muscettola, Nicola, Morris, HP, & Rajan, Kanna. 1999. Next generation remote agent planner. *Page 363 of : Artificial Intelligence, Robotics and Automation in Space*, vol. 440.

Karp, P D., Myers, K L., & Gruber, T. 1995. The generic frame protocol. *Pages 768–774 of : Proceedings of the International Joint Conference on Artificial Intelligence*.

Kichkaylo, Tatiana, van Buskirk, Chris, Singh, Sameer, Neema, Himanshu, Orosz, Michael, & Neches, Robert. 2007. Mixed-Initiative Planning for Space Exploration Missions. *In : ICAPS Workshop on Moving Planning and Scheduling Systems into the Real World*.

Klusch, M., Gerber, A., & Schmidt, M. 2005. Semantic web service composition planning with owls-xplan. *Pages 55–62 of : Proceedings of the Association for the Advancement of Artificial Intelligence Fall Symposium on Agents and the Semantic Web*.

Koehler, J., & Hoffmann, J. 1999. *Handling of inertia in a planning system.* Tech. rept.

Kovacs, D L. 2011. *BNF definition of PDDL 3.1.* Tech. rept.

Kuter, U., Nau, D. S., Pistore, M., & Traverso, P. 2009. Task decomposition on abstract states, for planning under nondeterminism. *Artificial Intelligence*, **173**(5-6), 669–695.

Lallement, R., De Silva, L., & Alami, R. 2014. HATP : An HTN planner for robotics. *In : In proceedings of the International Conference on Automated Planning and Scheduling Workshop on Planning and Robotics*.

Long, D., & Fox, M. 2003. The 3rd international planning competition : Results and analysis. *Journal of Artificial Intelligence Research*, **20**, 1–59.

Maldague, P, Ko, A, Page, D, & Starbird, T. 1998. APGEN : A multi-mission semi-automated planning tool. *Pages 363–365 of : First International NASA Workshop on Planning and Scheduling*.

Marconi, Lorenzo, Melchiorri, Claudio, Beetz, Michael, Pangercic, Dejan, Siegwart, Roland, Leutenegger, Stefan, Carloni, Raffaella, Stramigioli, Stefano, Bruyninckx, Herman, Doherty, Patrick, *et al.* . 2012. The SHERPA project : Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments. *Pages 1–4 of : Safety, Security, and Rescue Robotics (SSRR), 2012 IEEE International Symposium on.* IEEE.

Marthi, B., Russell, S J., & Wolfe, J. 2008. Angelic Hierarchical Planning : Optimal and Online Algorithms. *Pages 222–231 of : Proceedings of the International Conference on Automated Planning and Scheduling.*

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., & Wilkins, D. 1998. PDDL-the planning domain definition language.

Mistry, P., Maes, P., & Chang, L. 2009. WUW-wear Ur world : a wearable gestural interface. *Pages 4111–4116 of : CHI'09 extended abstracts on Human factors in computing systems.* ACM.

Montreuil, Vincent, Clodic, Aurélie, Ransan, Maxime, & Alami, Rachid. 2007. Planning human centered robot activities. *Pages 2618–2623 of : Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on.* IEEE.

Muñoz Avila, Héctor, Aha, David W., Breslow, Len, & Nau, Dana. 1999. HICAP : An Interactive Case-based Planning Architecture and Its Application to Noncombatant Evacuation Operations. *Pages 870–875 of : Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence.* AAAI '99/IAAI '99. Menlo Park, CA, USA : American Association for Artificial Intelligence.

Mulvehill, Alice M. 1996. Building, remembering, and revising force deployment plans. *Advanced Planning Technology*, 201–205.

Myers, K L., Tyson, W M., Wolverton, M J., Jarvis, P A., Lee, T J., & desJardins, M. 2002. PASSAT : A user-centric planning framework. *Pages 1–10 of : Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space.*

Myers, K L., Jarvis, P., Tyson, M., & Wolverton, M. 2003. A Mixed-initiative Framework for Robust Plan Sketching. *Pages 256–266 of : International Conference on Automated Planning and Scheduling.*

Nau, D., Cao, Y., Lotem, A., & Munoz-Avila, H. 1999. SHOP : Simple hierarchical ordered planner. *Pages 968–973 of : Proceedings of the international joint conference on Artificial intelligence.* Morgan Kaufmann Publishers Inc.

Nau, D., Cao, Y., Lotem, A., & Munoz-Avila, H. 2006. *SHOP and M-SHOP : Planning with ordered task decomposition*. Tech. rept. MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE.

Nau, D. S, Au, T., Ilghami, O., Kuter, U., Murdock, J M., Wu, D., & Yaman, F. 2003. SHOP2 : An HTN planning system. *Journal of Artificial Intelligence Research*, **20**, 379–404.

Novick, D G., & Sutton, S. 1997. What is mixed-initiative interaction? *Pages 114–116 of : Procedings of the AAAI Spring Symposium on Computational Models for Mixed Initiative Interaction*.

Paik, I., & Maruyama, D. 2007. Automatic web services composition using combining HTN and CSP. *Pages 206–211 of : Proceedings of the International Conference on Computer and Information Technology*. IEEE.

Pavel, S., & Roman, B. 2005. Encoding HTN Planning as a Dynamic CSP. *Page 868 of : Proceedings of the International Conference on Principles and Practice of Constraint Programming*.

Pednault, E. 1988. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence*, **4**(3), 356–372.

Pednault, E. 1989. ADL : Exploring the Middle Ground Between STRIPS and the Situation Calculus. *Kr*, **89**, 324–332.

Pellier, D., & Fiorino, H. 2007. A Unified Framework Based on HTN and POP Approaches for Multi-Agent Planning. *Pages 285–288 of : Proceedings of the International Conference on Intelligent Agent Technology*.

Pellier, D., & Fiorino, H. 2018. PDDL4J : a planning domain description library for java. *Journal of Experimental & Theoretical Artificial Intelligence*, **30**(1), 143–176.

Penberthy, J S., Weld, D S., *et al.* . 1992. UCPOP : A Sound, Complete, Partial Order Planner for ADL. *Kr*, **92**, 103–114.

Ramoul, A., Pellier, D., Fiorino, H., & Pesty, S. 2016. HTN Planning Approach Using Fully Instantiated Problems. *Pages 113–120 of : Proceeding of the International Conference on Tools in Artificial Intelligence*. IEEE.

Rao, Anand S, & Georgeff, Michael P. 1991. Deliberation and its role in the formation of intentions. *Pages 300–307 of : Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc.

Sacerdoti, E. 1975a. The nonlinear nature of plans. *Pages 206–214 of : Proceedings of the International Joint Conference on Artificial Intelligence*.

Sacerdoti, E. 1975b. *A structure for plans and behavior*. Tech. rept. SRI International Menlo Park CA Artificial Intelligence Center.

Scapin, D L., & Bastien, J M. 1997. Ergonomic criteria for evaluating the ergonomic quality of interactive systems. *Behaviour & Information Technology*, **16**(4-5), 220–231.

Shivashankar, V., Alford, R., Kuter, U., & Nau, D S. 2013. The GoDeL Planning System : A More Perfect Union of Domain-Independent and Hierarchical Planning. *Pages 2380–2386 of : Proceedings of the International Joint Conference on Artificial Intelligence*.

Sirin, E., Parsia, B., Wu, D., Hendler, J., & Nau, D. 2004. HTN planning for web service composition using SHOP2. *Web Semantics : Science, Services and Agents on the World Wide Web*, **1**(4), 377–396.

Smith, David E. 2012. Planning as an iterative process. *In : Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Smith, R W. 1993. Effective spoken natural language dialog requires variable initiative behavior : an empirical study. *Pages 101–106 of : the 1993 AAAI Fall Symposium on Human-Computer Collaboration : Reconciling Theory, Synthesizing Practice*.

Sohrabi, S., Baier, J A., & McIlraith, S A. 2009. HTN Planning with Preferences. *Pages 1790–1797 of : Proceedings of the International Joint Conference on Artificial Intelligence*.

Tate, A. 1976. *Project planning using a hierarchic non-linear planner*. Department of Artificial Intelligence, University of Edinburgh.

Tate, A. 1977. Generating project networks. *Pages 888–893 of : Proceedings of the International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc.

Tate, A., Drabble, B., & Kirby, R. 1994. O-Plan2 : an open architecture for command, planning and control. *In : Proceedings of the Intelligent Scheduling*.

Tecuci, G., Boicu, M., & Cox, M T. 2007. Seven aspects of mixed-initiative reasoning : An introduction to this special issue on mixed-initiative assistants. *Artificial Intelligence Magazine*, **28**(2), 11.

Veloso, Manuela, Carbonell, Jaime, Perez, Alicia, Borrajo, Daniel, Fink, Eugene, & Blythe, Jim. 1995. Integrating planning and learning : The PRODIGY architecture. *Journal of Experimental & Theoretical Artificial Intelligence*, **7**(1), 81–120.

Veloso, Manuela M, Mulvehill, Alice M, & Cox, Michael T. 1997. Rationale-supported mixed-initiative case-based planning. *Pages 1072–1077 of : AAAI/IAAI*.

Walker, M., & Whittaker, S. 1990. Mixed initiative in dialogue : An investigation into discourse segmentation. *Pages 70–78 of : Proceedings of the 28th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Weser, M., Off, D., & Zhang, J. 2010. HTN robot planning in partially observable dynamic environments. *Pages 1505–1510 of : Proceedings of the International Conference on Robotics and Automation*. IEEE.

Wilkins, D E. 1984. Domain-independent planning Representation and plan generation. *Artifical Intelligence Journal*, **22**(3), 269–301.

Wilkins, D. E. 1990. Can AI planners solve practical problems? *Computational intelligence Journal*, **6**(4), 232–246.

Wu, D., Parsia, B., Sirin, E., Hendler, J., & Nau, D. 2003. Automating DAML-S web services composition using SHOP2. *Pages 195–210 of : International Semantic Web Conference*. Springer.